1971

# Computational aspects of geometric programming

Gary K. Oleson
*Iowa State University*

71-26,877

OLESON, Gary K., 1940-
COMPUTATIONAL ASPECTS OF GEOMETRIC PROGRAMMING.

Iowa State University, Ph.D., 1971
Computer Science

University Microfilms, A XEROX Company, Ann Arbor, Michigan

# Computational aspects of geometric programming

by

Gary K. Oleson

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of
The Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major Subjects: Economics
Computer Science

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

Heads of Major Departments

Signature was redacted for privacy.

Dean of Graduate College

Iowa State University
Of Science and Technology
Ames, Iowa

1971

# TABLE OF CONTENTS

# INTRODUCTION

Mathematical programming concerns the optimization of some objective function subject to constraints. In economics, its application is the optimal allocation of resources. The determination of an optimal product mix is a manufacturing problem. The least cost, most profitable, most reliable, or even feasible system design of a chemical reactor is a chemical engineering endeavor. Thus, it can be seen that many and varied application areas exist.

The efficient numerical solution of many large mathematical programming problems has been possible only since the introduction of programmed digital computers. A large effort has been expended on development of algorithms for special classes or structures of mathematical programming. Linear programming is the most used and developed of the mathematical programming family. Linear programming (5) is a method of solving problems that have linear objective functions and linear constraints. Non-linear programming (4, 6, 11, 12) generally concerns problems which are not totally linear. One method of solution of non-linear programs is to approximate the non-linearity and solve a linear program. This method is utilized in Chapter 3.

Geometric programming and its extensions treat a class of non-linear problems that are expressed in terms of polynomials. Transformation of a geometric program yields a mathematical

program with linear constraints. In special cases, the result is a linear independent system with a unique solution. Chapter 2 is a survey of developments that lead up to the linearization of geometric programs.

Chapter 3 deals with a new computation method associated with a linearization of geometric and general polynomial programs. A general polynomial program is transformed to a geometric program with a monomial objective function and monomial constraints which are restricted to being active or tight at the optimal. The degree of difficulty is not decreased in the transformation. This transformation forms a class of lineer programs. The weights used in the linearization appear only in the objective function of the formed linear program. This feature was used to develop a computation method which parametrically changes the objective function until all constraints are active or tight. Formal proofs are not given for the convergence to global optimums. The linearization and computation of optimums for two examples from current literature (1, 9) is given. Convergence to a global optimum was obtained in these cases. However, it should be emphasized that the mathematical theory of global convergence is not worked out for arbitrary and general cases.

The last chapter is a summary of results and ideas for future research.

## HISTORICAL DEVELOPMENT

The general statement of a mathematical program is to determine a vector x that solves the problem:

minimize $f(x)$,

subject to

$$g_i(x) \leq 0, \quad i = 1,\ldots,m,$$

$$h_i(x) = 0, \quad i = 1,\ldots,p.$$

The above problem can be solved in the general case by the classical Lagrange multiplier technique (3, 5, 12). The Lagrange technique is to formulate a composite function of the objective function and the constraints. The constraints are required to be of the form

$$h_i(x) = 0.$$

A constraint not of this form can meet this requirement by the addition of a slack variable

$$h_i(x) = g_i(x) - X_s = 0,$$

where

$$g_i(x) \leq 0,$$

$$X_s \leq 0.$$

Then, with no loss of generality, the problem is

minimize $f(x)$,

subject to

$$g_i(x) = 0, \quad i = 1,\ldots,m.$$

The composite function then is of the form

$$L(x,\lambda) = f(x) - \sum_{i=1}^{m} \lambda_i (g_i(x)).$$

The added variables $\lambda_i$ are called Lagrange multipliers.

In that all $g_i(x) = 0$ (original constraints), the composite function $L(x,\lambda) = f(x)$. Let $(x,\lambda)$ satisfy the composite function. It can be shown that if $(x,\lambda)$ is a local or global minimum or maximum, then x is a corresponding critical point for the original problem. The problem is thus reduced to analyzing $L(x,\lambda)$ by some unconstrained optimization method.

A critical point corresponds to a point where all of the partial derivatives of the composite function are equal to zero:

$$\frac{\partial L}{\partial x_i} = \frac{\partial f}{\partial x_i} - \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(x)}{\partial x_i} = 0, \qquad i = 1,2,\ldots,n,$$

$$\frac{\partial L}{\partial \lambda_i} = -g_i(x) = 0, \qquad i = 1,2,\ldots,m.$$

The critical points are obtained by solving this set of (m + n) equations. Further analysis of the critical points using Kuhn-Tucker conditions (12) is required to determine the local optimality of an extreme point. The determination of a global optimum is difficult, if possible, when many local critical points exist.

The Lagrange multiplier technique is not a practical or powerful computation method. It is a formidable task to solve the set of resulting equations (many times non-linear). The

number of critical points to be examined may be so large that it is impractical to attempt identifying a global minimum or maximum even if such exists.

Mathematical programming methods have been developed to overcome the computational difficulty of directly applying the Lagrange multiplier method. Fiacco and McCormick (4) state, though, that the Lagrange method is inextricably associated with every method for mathematical programming. Their method of solving the problem is called sequential unconstrained minimization technique (SUMT).

## SUMT

Fiacco's and McCormick's (4) programming method called SUMT (sequential unconstrained minimization technique) treats Lagrange multipliers as a special case of their penalty and barrier functions.

The general form of the SUMT method is

minimize $f(x)$,

subject to

$$g_i(x) \geq 0, \quad i = 1,2,\ldots,m_1,$$

$$h_i(x) \geq 0, \quad i = m_1+1,\ldots,m.$$

Define an unconstrained auxiliary function of the form

$$V(x,r_k^{(1)},r_k^{(2)}) = f(x) + s(r_k^{(1)})G(x) + p(r_k^{(2)})H(x).$$

In the special case where $s(r_k^{(1)})$ and $p(r_k^{(2)})$ are constants and $G(x) = x$ and $H(x) = x$, the auxiliary function is the

Lagrange composite function. If a minimum exists in a compact set containing every limit point of any subsequence $(x^k)$ of the minimizing points, the following hold:

$$\text{(i)}^{\circ} \quad \lim_{k \to \infty} s(r_k^{(1)})G(x^k) = 0,$$

$$\text{(ii)} \quad \lim_{k \to \infty} p(r_k^{(2)})H(x^k) = 0,$$

$$\text{(iii)} \quad \lim_{k \to \infty} V(x^k, r_k^{(1)}, r_k^{(2)}) = V^*,$$

$V^*$ is the optimal,

k is the iteration number,

$s(r)G(x)$ is a penalty function,

$p(r)H(x)$ is a barrier function.

The algorithm starts with an initial $r > 0$ and a vector x.

The computation advantage of the SUMT method is that any unconstrained minimization technique can be applied to the auxiliary function. Also, constraints that are satisfied at any iteration can be dropped. Clearly, the computation is easier than directly applying the Lagrange multiplier method. Most mathematical programming methods use a method of optimization that is characterized by the problem structure--linear programming is an example. Other methods transform the original problem to another form for computation advantages. Geometric programming, like SUMT, uses a transformed problem.

## Linear Programming

Linear programming (5) is a special mathematical program where the objective function and all of the constraints are

linear. The computation efficiency of linear programming is used later as a basis for geometric programming. The general form of the problem is to

minimize $c^t x$,

subject to

$Ax = b$, A is m x n,

$x_i \geq 0$, $i = 1,\ldots,n$.

The linear constraints of $Ax = b$ form a convex set of possible values of x. It can be shown that a linear objective function attains its optimal at one of the extreme points of the convex set. Each extreme point of this convex set can be represented by the vector x having at most m components greater than zero.

Dantzig in 1947, as mentioned in Hadley, (5, p. 20) developed the simplex method of linear programming. The simplex method generates a new extreme point satisfying

$Ax = b$,

$x \geq 0$,

at each iteration. If this extreme point is optimal, computation is terminated. If not optimal, adjacent extreme points are considered that may lead to the optimal. The optimal vector x has components $x_i \geq 0$ for all i corresponding to the independent columns of A used to find x (an extreme point). All other components equal zero.

The simplex method thus reduces the optimization process to examination of an extreme point and adjacent points at each

iteration. In that only a finite number of extreme points must be considered, the method is convergent. The simplex method obtains its efficiency by not requiring the examination of all extreme points.

Whenever the linear restriction is removed from the general problem, a non-linear program is the result. Geometric programming is one type of non-linear programming.

## Geometric Programming

Geometric programming handles the class of functions which have been termed posynomials by Duffin, Peterson, and Zener (3)--posynomials being polynomials with the restriction that all coefficients are positive. Geometric programming is a technique that finds the optimal value of the objective function with a dual problem. The primal variables are determined from the optimal term allocation which is given by the dual solution. The name geometric programming is derived from its initimate connection with geometric concepts, the most important being orthogonality of vectors.

The minimization of polynomials uses the geometric inequality which states that the arithmetic mean is at least as great as the geometric mean. As an illustration:

$$\tfrac{1}{2}U_1 + \tfrac{1}{2}U_2 \geq U_1^{\frac{1}{2}}U_2^{\frac{1}{2}},$$

$$U_1 \geq 0, \qquad U_2 \geq 0,$$

note

$$(U_1 - U_2)^2 \geq 0,$$

$$U_1^2 - 2U_1U_2 + U_2^2 \geq 0,$$

add $4U_1U_2$ to both sides

$$U_1^2 + 2U_1U_2 + U_2^2 \geq 4U_1U_2,$$

taking the square root and dividing by 2 yields

$$\tfrac{1}{2}U_1 + \tfrac{1}{2}U_2 \geq U_1^{\frac{1}{2}}U_2^{\frac{1}{2}}.$$

It can be noted that strict equality holds only when $U_1 = U_2$.

A more general form of the inequality is

$$\delta_1U_1 + \delta_2U_2 + \ldots + \delta_tU_t \geq U_1^{\delta_1}U_2^{\delta_2},\ldots,U_t^{\delta_t},$$

$$U_1 \geq 0,$$

$$\delta_1 \geq 0,$$

$$\sum_{t=1}^{r} \delta_i = 1.$$

A change of variables $u_i = \delta_iU_i$ yields

$$u_1 + u_2 + \ldots + u_t \geq \left(\frac{u_1}{\delta_1}\right)^{\delta_1}\left(\frac{u_2}{\delta_2}\right)^{\delta_2},\ldots,\left(\frac{u_t}{\delta_t}\right)^{\delta_t}.$$

Let the left hand side of the inequality correspond to a posynomial:

$$g(x) = u_1 + u_2 + \ldots + u_t,$$

$$u_i = c_ix_1^{a_{i1}}x_2^{a_{i2}},\ldots,x_n^{a_{in}}, \qquad i = 1,\ldots,t.$$

Let T equal the number of terms in the posynomial and N equal the number of variables in the posynomial. Let V correspond to the right hand side of the above inequality, thus

$$g \geq V,$$

or

$$V(\delta,x) = \left(\frac{c_1}{\delta_1}\right)^{\delta_1}\left(\frac{c_2}{\delta_2}\right)^{\delta_2},\ldots,\left(\frac{c_T}{\delta_T}\right)^{\delta_T} x_1^{D_1} x_2^{D_2},\ldots,x_n^{D_n},$$

where the $D_j$ are

$$D_j = \sum_{i=1}^{T} \delta_i a_{ij}, \qquad j = 1,2,\ldots,N.$$

Now if the $\delta_i$ are chosen so that $V(\delta,x)$ does not depend on x, let

$$D_j = 0, \qquad j = 1,2,\ldots,N,$$

then $V(\delta,x)$ reduces to

$$V = \left(\frac{c_1}{\delta_1}\right)^{\delta_1}\left(\frac{c_2}{\delta_2}\right)^{\delta_2},\ldots,\left(\frac{c_T}{\delta_T}\right)^{\delta_T}.$$

It can be shown (3) that g has a positive lower bound M and M is also a least upper bound of V. Thus

$$g(x) \geq M \geq V(\delta).$$

Now it will be shown that

$$g(x) = V(\delta),$$

at the optimal minimum value of g(x).

Geometric programs are assumed to have a minimum at a point where

$$x_j > 0, \qquad j = 1,2,\ldots,N.$$

For a minimizing point $\underline{x}$, the derivatives of $g(x)$ must equal zero, so

$$\frac{\partial g(\underline{x})}{\partial x_j} = 0, \qquad j = 1,2,\ldots,N,$$

or

$$x_j\left(\frac{\partial g(\underline{x})}{\partial x_j}\right) = \sum_{i=1}^{T} x_j\left(\frac{\partial u_i(\underline{x})}{\partial x_j}\right) = \sum_{i=1}^{T} u_i(\underline{x})a_{ij} = 0.$$

Dividing these equations by $g(\underline{x})$ and defining

$$\delta_i = \frac{u_i(\underline{x})}{g(\underline{x})}, \qquad i = 1,2,\ldots,T, \tag{1.1}$$

thus

$$\sum_{i=1}^{T} \delta_i a_{ij} = 0, \qquad j = 1,2,\ldots,N. \tag{1.2}$$

Thus, the vector $\underline{\delta}$ satisfies the orthogonality condition. From (1), the vector $\underline{\delta}$ satisfies the normality condition, which means

$$g(\underline{x}) = g(\underline{x})^{\delta_1} \cdot g(\underline{x})^{\delta_2},\ldots,g(\underline{x})^{\delta_T},$$

and from (1) also

$$g(x) = \left(\frac{u_1(\underline{x})}{\delta_1}\right)^{\delta_1},\ldots,\left(\frac{u_T(\underline{x})}{\delta_T}\right)^{\delta_T}.$$

It follows that

$$g(\underline{x}) = V(\delta).$$

The preceding can be extended to include inequality constraints. A more general form of the geometric inequality is to let $\Delta_n$ be the unnormalized weights and $\lambda$ be their sum:

$$\lambda = \Delta_1 + \Delta_2 + \ldots + \Delta_T,$$

let the relation hold

$$\Delta_i = \lambda\delta_i, \qquad i = 1,2,\ldots,T.$$

Substitution of $\delta_i$ by $\lambda\delta_i$ into the geometric inequality gives

$$u_1 + u_2 + \ldots + u_t \geq \left(\frac{u_1}{\Delta_1}\right)^{\frac{\Delta_1}{\lambda}} \left(\frac{u_2}{\Delta_2}\right)^{\frac{\Delta_2}{\lambda}}, \ldots, \left(\frac{u_t}{\Delta_t}\right)^{\frac{\Delta_t}{\lambda}}.$$

Taking both sides to the $\lambda$ power yields

$$(u_1 + u_2 + \ldots + u_t)^{\lambda} \geq \left(\frac{u_1}{\Delta_1}\right)^{\Delta_1} \left(\frac{u_2}{\Delta_2}\right)^{\Delta_1} \lambda^{\lambda}.$$

Now if a constraint is of the form

$$g_k(x) \leq 1,$$

it can be converted to the product form of

$$1 \geq g_k(x) \geq \left(\frac{u_{1k}}{\Delta_{1k}}\right)^{\Delta_1} \left(\frac{u_{2k}}{\Delta_{2k}}\right)^{\Delta_{2k}} \left(\frac{u_{Tk}}{\Delta_{Tk}}\right)^{\Delta_{Tk}} \lambda_1^{\lambda_1}.$$

The objective function being of the form

$$g_o^{\lambda_o} \geq \left(\frac{u_1}{\Delta_1}\right)^{\Delta_1}, \ldots, \left(\frac{u_T}{\Delta_T}\right)^{\Delta_T} \lambda_o^{\lambda_o}.$$

Multiplication of the extreme sides of both inequalities yields

$$g_o^{\lambda_o} \geq \left(\frac{u_1}{\Delta_1}\right)^{\Delta_1}, \ldots, \left(\frac{u_T}{\Delta_T}\right)^{\Delta_T} \left(\frac{u_{1k}}{\Delta_{1k}}\right)^{\Delta_{1k}}, \ldots, \left(\frac{u_{Tk}}{\Delta_{Tk}}\right)^{\Delta_{Tk}} \lambda_o^{\lambda_o} \lambda_1^{\lambda_1}.$$

In that the inequality holds for any $\Delta_i$, it is convenient to set $\lambda_o = 1$. Now denoting the weights normalized in this fashion as $\delta_i$, the combined system can be expressed as

$$g_o(x) \geq \left(\frac{u_1}{\delta_1}\right)^{\delta_1},\ldots,\left(\frac{u_t}{\delta_t}\right)^{\delta_t}\left(\frac{u_{1k}}{\delta_{1k}}\right)^{\delta_{1k}},\ldots,\left(\frac{u_{Tk}}{\delta_{Tk}}\right)^{\delta_{Tk}}\lambda_k^{\lambda_k}.$$

A dual function $V(\delta,x)$ is of the form

$$V(\delta,x) = \left(\frac{c_1}{\delta_1}\right)^{\delta_1},\ldots,\left(\frac{c_t}{\delta_t}\right)^{\delta_t}\left(\frac{c_{1k}}{\delta_{1k}}\right)^{\delta_{1k}},\ldots,\left(\frac{c_{Tk}}{\delta_{Tk}}\right)^{\delta_{Tk}}$$

$$\lambda_k^{\lambda_k} x_1^{D_1} x_2^{D_1} x_n^{D_n},$$

where

$$D_i = \sum_{i=1}^{T+T_k} \delta_j.$$

Selecting $\delta_i$ so that

$$D_j = 0, \qquad j = 1,\ldots,N,$$

the dual reduces to

$$V(\delta) = \left(\frac{c_1}{\delta_1}\right)^{\delta_1},\ldots,\left(\frac{c_t}{\delta_t}\right)^{\delta_t}\left(\frac{c_{1k}}{\delta_{1k}}\right)^{\delta_{1k}},\ldots,\left(\frac{c_{Tk}}{\delta_{Tk}}\right)^{\delta_{Tk}}\lambda_k^{\lambda_k},$$

then

$$g_o(x) \geq M \geq u(\delta),$$

M giving a lower bound on $g_o(\underline{x})$.

Computationally, the transformed problem or dual is the easiest to solve, which is maximize $u(\delta)$ subject to a linear normality equation and the orthogonality conditions (one for each variable). It can be seen that, if the total number of terms (a $\delta_i$ associated with each) - total number of variables (one equation for each) - 1 (a normality equation) equals

zero, the solution is unique. Duffin calls the above relation the degree of difficulty (3).

Define $M + 1$ generalized polynomials $g_o(x)$, $g_1(x)$, and $g_m(x)$, each consisting of a variable number of terms designated $T_m$ for $m = 0,1,\ldots,M$. By appending the index $m$ to every term and exponent of the $M + 1$ posynomials, it is possible to formulate the primal geometric program:

minimize $g_o(x)$,

subject to

$$g_m(x) \leq 1, \qquad m = 1,2,\ldots,M,$$

where

$$g_m(x) = \sum_{t=1}^{T_m} g_m^t(x) = \sum_{t=1}^{T_m} c_{mt} \prod_{n=1}^{N} x_n^{a_{mnt}}, \qquad m = 0,1,\ldots,M,$$

with

$$c_{mt} > 0, \qquad t = 1,\ldots,T_m, \qquad m = 0,1,\ldots,M,$$

$$0 < x_n < \infty, \qquad n = 1,\ldots,N.$$

Exponents $a_{mnt}$ are arbitrary real numbers.

The dual of the primal geometric program is

$$\text{maximize } V(\delta) = \prod_{m=o}^{M} \prod_{t=1}^{T_m} \left(\frac{c_{mt}\delta_{mo}}{\delta_{mt}}\right)^{\delta_{mt}},$$

where

$$\delta_{oo} = 1$$

$$\lim_{\delta_{mt} \to 0} \frac{c_{mt}\delta_{mo}}{\delta_{mt}} = 1, \qquad m = 0,1,\ldots,M, \qquad t = 1,\ldots,T_m,$$

subject to linear constraints, the normality equation is

$$\sum_{t=1}^{T_o} \delta_{ot} = 1,$$

the orthogonal equations are

$$\sum_{m=o}^{M} \sum_{t=1}^{T_m} a_{mnt}\delta_{mt} = 0, \qquad n = 1,\ldots,N,$$

$$\alpha_{mo} = \sum_{t=1}^{T_m} \delta_{mt}, \qquad m = 1,\ldots,M,$$

and the non-negative restriction

$$\delta_{mt} \geq 0, \qquad m = 0,1,\ldots,M, \qquad t = 0,1,\ldots,T_m.$$

The main theorem of geometric programming is due to Duffin, Peterson, and Zener (3) and is given below:

THEOREM 1-1 (3) If x satisfies the constraints of the primal program and $\delta$ satisfies the constraints of the dual program, then

$$g_o(x) \geq v(\delta).$$

Moreover, under the same conditions, $g_o(x^o) = v(\delta^o)$ if, and only if,

$$\delta^o_{ot} = \frac{g^t_o(x)}{g_o(x)} = \frac{c_{ot} \prod_{n=1}^{M} (x^o_n)^{a_{ont}}}{g_o(x^o)}, \qquad t = 1,\ldots,T_o,$$

and for $\delta_{mo} > 0$,

$$\frac{\delta^o_{mt}}{\delta^o_{mo}} = g^t_m(x^o) = c_{mt} \prod_{n=1}^{M} (x^o_n)^{a_{mnt}}, \qquad \begin{matrix} m = 1,\ldots,M, \\ t = 1,\ldots,T_m. \end{matrix}$$

The main theorem provides the fundamental relationships between the primal and dual variables at the optimum. As each dual variable determines a value of each primal term, the computation of the primal variables involves solving a linear system with the logarithm of the primal variables as unknowns.

The dual program has a computational advantage in that all of its constraints are linear. Also, it has been proven that the log $(V(\delta))$ is concave and has the same maximizing point as $V(\delta)$. Thus, the global dual maximum that is computed corresponds to a global minimum point of the primal.

## Generalized Polynomial Programs

Passy (10) extended geometric programs to include the general polynomial functions. These result when the positive coefficient restriction on all terms is relaxed. The introduction of a signum function $\sigma_t$ with every term yields a program very similar to a geometric program.

Using the same geometric program conventions, the general polynomial primal program is

minimize $g_0(x)$,

subject to

$$g_m(x) \leq \sigma_m (= \pm 1), \qquad m = 1, \ldots, M,$$

where

$$g_m(x) = \sum_{t=1}^{T_m} g_m^t(x) = \sum_{t=1}^{T_m} \sigma_{mt} c_{mt} \prod_{n=1}^{N} x_n^{a_{mnt}},$$

$$m = 0, 1, \ldots, M,$$

with

$$\sigma_{mt} = \pm 1, \qquad t = 1,\ldots,T_m, \qquad m = 0,1,\ldots,M,$$

such that

$$c_{mt} > 0, \qquad t = 1,\ldots,T_m, \qquad m = 0,1,\ldots,M,$$

and

$$0 < x_n < \infty, \qquad n = 1,\ldots,N.$$

A constraint signum function $\sigma_m$ is introduced for convenience. It is 1 or -1 depending on how a constraint is written.

Corresponding to the above primal, there exists a quasidual program which is defined as

pseudomaximize $z(\delta)$,

where

$$z(\delta) = \sigma_o \left( \prod_{m=0}^{M} \prod_{t=1}^{T_m} \left( \frac{c_{mt}\delta_{mo}}{\delta_{mt}} \right)^{\sigma_{mt}\delta_{mt}} \right)^{\sigma_o},$$

where

$$\sigma_o = \text{sign}\,(g_o(\underline{x}^o)), \qquad \delta_{oo} = 1,$$

$$\lim_{\delta_{mt} \to 0} \frac{c_{mt}\delta_{mo}}{\delta_{mt}} = 1, \qquad m = 0,1,\ldots,M, \qquad t = 1,\ldots,T_m,$$

subject to the linear constraints

$$\sum_{t=1}^{T_o} \sigma_{ot}\delta_{ot} = \sigma_o,$$

$$\sum_{m=0}^{M} \sum_{t=1}^{T_m} \sigma_{mt} a_{mnt} \delta_{mt} = 0, \qquad n = 1,2,\ldots,N,$$

$$\delta_{mo} = \sigma_m \sum_{t=1}^{T_m} \sigma_{mt} \delta_{mt}, \qquad m = 1,2,\ldots,M,$$

and the non-negative conditions

$$\delta_{mt} \geq 0, \qquad m = 0,1,\ldots,T_m,$$

and a constraint qualification

$$\delta_{mt} \neq 0 \text{ if, and only if, } \delta_{mo} \neq 0, \qquad m = 1,\ldots,M.$$

For generalized polynomial programming, a weaker type of duality is exhibited. Let x be the set of primal variables that satisfy the primal program and $\Delta$ the set of dual variables $\delta$ satisfying the dual program constraints. Using these definitions, the following theorem is due to Passy (10):

THEOREM 1-2 If $x^o$ is in x and is a minimum of the primal program, such that

$$g_o(x) \geq g_o(x^o),$$

for all $(x - x^o) < \epsilon$ where $\epsilon$ is a real positive scalar, there exists a set of dual variables $\delta^o$ in $\Delta$ such that

$$z(\delta^o) = g_o(x^o).$$

Moreover, the Kuhn-Tucker (8) necessary conditions for a maximum of the dual program are satisfied at $\delta^o$, that is

$$\frac{\partial L(\delta^o, \lambda^o)}{\partial \sigma_{mt}} = 0, \qquad \text{if } \delta^o_{mt} > 0, \qquad m = 0,1,\ldots,M, \quad t = 1,\ldots,T_m,$$

$$\frac{\partial L(\delta^o, \lambda^o)}{\partial \delta_{mt}} \leq 0, \qquad \text{if } \delta^o_{mt} = 0, \qquad m = 0,1,\ldots,M, \quad t = 1,\ldots,T_m,$$

$$\frac{\partial L(\delta^o,\lambda^o)}{\partial \lambda_i} = 0, \qquad i = 0,1,\ldots,N+M,$$

where the Lagrangian function of the dual program is given by

$$L(\delta,\lambda) = z(\delta) + \lambda_o\Big(\sum_{t=1}^{T_o} \sigma_{ot} + \delta_{ot} - \sigma_o\Big)$$

$$+ \sum_{n=1}^{N} \lambda_n \Big(\sum_{m=o}^{M} \sum_{t=1}^{T_m} \sigma_{mt} a_{mnt} \delta_{mt}\Big)$$

$$+ \sum_{m=1}^{M} \lambda_{N+M}\Big(\delta_{no} - \sigma_m \sum_{t=1}^{T_m} \sigma_{mt}\delta_{mt}\Big);$$

and $\lambda_0,\lambda_1,\ldots,\lambda_m$ are the Lagrange multipliers. Also, $z(\delta^o) = g_o(x)^o$ for $x^o$ in $x$ and $\delta^o$ in $\Delta$ if, and only if,

$$\delta^o_{ot} = \frac{g^t_o(x^o)}{\sigma_o g_o(x^o)} = \frac{c_{ot} \prod_{n=1}^{N} (x^o_n)^{a_{ont}}}{\sigma_o g_o(\underline{x})^o}, \qquad t = 1,\ldots,T_o,$$

and for $\delta^o_{mo} > 0$

$$\frac{\delta^c_{mt}}{\delta^o_{mo}} = g^t_m(\underline{x}^o) = c_{mt} \prod_{n=i}^{N} (x^o_n)^{a_{mnt}}, \qquad m = 1,\ldots,M,$$
$$t = 1,\ldots,T_m.$$

Duality theory in mathematical programming commonly has two aspects:

(a) The primal program is a constrained minimization program, and the dual program is a constrained maximization problem.

(b) The existence of a solution to one of these programs ensures the existence of a solution to the other,

in which case their evaluated objective functions are equal.

The geometric program theorem verifies both aspects. The generalized polynomial program theorem does not satisfy (a) in that the dual is not a maximization problem. The term pseudomaximization is used instead and is defined as the searching for a stationary point that satisfies the dual program. Generalized polynomial programs satisfy (b) in that a stationary point of the dual ensures a stationary point exists for the primal and the evaluated objective functions of both are equal. In summary, the solution of the dual generalized polynomial program does not in general yield a global minimum to the primal. A local minimum or maximum or saddle-point may be the result.

Although generalized polynomial dual programs are not constrained maximization problems, they do have the computation advantage of linear constraints. The following transformation of the dual objective function

$$d(\delta) = \sigma_o \log(\sigma_o z(\delta))$$

also helps in computation. Passy (10) has shown the above to be valid for all $\underline{\delta}$ in $\Delta$ near $\underline{\delta}^o$ and that the psuedomaximizing point of $d(\delta)$ and $z(\delta)$ are identical. The transformed function $d(\delta)$ though is not in general concave as is the geometric programming case.

Blau (1), using the theoretical results of Passy (10), developed a computational algorithm for generalized polynomial

programs. Certain assumptions are made in the algorithm development:

(a) The constraint signum functions $\sigma_1, \sigma_2, \ldots, \sigma_m$ and sign of objective function $\sigma_o$ at the local minimum are known.

(b) In the primal Lagrangian function

$$L(x,\mu) = g_o(x) + \sum_{m=1}^{M} \mu_m (g_m(x) - \sigma_m),$$

the Lagrange multipliers $\mu_1^o, \ldots, \mu_m^o$ exist at the local minimum $\underline{x}^o$ and are strictly positive:

$$\mu_m^o > 0, \qquad m = 1, \ldots, M.$$

Aspect (b), when extended to include Kuhn-Tucker conditions, in essence means that all constraints are tight or active at the optimal point. Note, this is Blau's definition of <u>well-formulated</u>.

Blau's method then is to find a stationary point of the transformed quasidual objective function

$$d(\delta) = \sum_{m=o}^{M} \sum_{t=1}^{T_m} \sigma_{mt} \delta_{mt} \log\left(\frac{c_{mt}\delta_{mo}}{\delta_{mt}}\right),$$

subject to

$$\sum_{t=1}^{T_o} \sigma_{ot}\delta_{ot} = \sigma_o,$$

$$\sum_{n=o}^{M} \sum_{t=1}^{T_m} \sigma_{mt} a_{mnt} \delta_{mt} = 0, \qquad n = 1, \ldots, N,$$

$$\delta_{mo} - \sigma_m \sum_{t=1}^{T_m} \sigma_{mt}\delta_{mt} = 0, \qquad m = 1,\ldots,M,$$

and the positivity conditions

$$\delta_{mt} > 0, \qquad m = 0,1,\ldots,M, \qquad t = 0,1,\ldots,T_m,$$

where

$$T \equiv \sum_{m=o}^{M} T_m.$$

The terms of the transformed dual can be arranged to form

$$d(\delta) = \sum_{m=o}^{M} \sum_{t=1}^{T_m} \sigma_{mt}\delta_{mt}\log\left(\frac{c_{mt}}{\sigma_{mt}}\right) + \sum_{m=o}^{M} \log \delta_{mo} \sum_{t=1}^{T_m} \sigma_{mt}\delta_{mt}.$$

Substitution of linear constraints containing $\sigma_n$ yields

$$d(\delta) = \sum_{n=o}^{M} \sum_{t=1}^{T_m} \sigma_{mt}\delta_{mt}\log\left(\frac{c_{mt}}{\sigma_{mt}}\right) + \sum_{m=1}^{M} \sigma_m\delta_{mo}\log \delta_{mo}.$$

Introducing (M + N + 1) classical Lagrange multipliers, the Lagrangian function is formed. By definition, the partials of this function vanish at a stationary point. Thus, a (T + N + 2M + 1) non-linear system of equations must be solved. Blau takes advantage of the <u>separability</u> of the linear-logarithmic system; that given a vector of Lagrangian multipliers, the dual vector $\delta$ is completely determined. This feature allows an iterative method of solving the non-linear system requiring inversion of a (N + M + 1) matrix at each iteration. This computation task is significantly easier than inverting the (T + 2M + N + 1) matrix necessary at each

iteration for the classical Newton method of solution for the original system.

This method is restricted to finding a stationary point and requires a starting point for initialization. Blau has reported (1) that a stationary point is not obtained if poor approximizations are used for initialization.

## Linear Programs as Geometric Programs

Federowicz, in Duffin (3, p. 265), has shown that all linear programs can be expressed as geometric programs of a special type. A linear program can be expressed in general as

$$\text{minimize } G_o = a_{o1}z_1 + a_{o2}z_2 + \ldots + a_{on}z_n + C_o,$$

subject to

$$G_1(z) = a_{11}z_1 + \ldots + a_{1n}z_n + C_1 \leq 0,$$

$$=$$

$$=$$

$$G_m(z) = a_{m1}z_1 + \ldots + a_{mn}z_n + C_m \leq 0,$$

where $a_{ij}$ and $C_i$ are arbitrary constants.

Using the following transforms

$$G_i = \log g_i, \qquad C_i = \log c_i, \qquad z_i = \log x_i,$$

and the monotonicity of the logarithm function permits the resulting geometric program:

$$\text{minimize } g_o(x) = c_i x_1^{a_{o1}}, \ldots, x_n^{a_{on}},$$

subject to

$$x_i > 0, \quad i = 1,2,\ldots,n,$$

$$g_i(x) = c_i x_1^{a_{i1}},\ldots,x_n^{a_{in}} \leq 1, \quad i = 1,2,\ldots,m.$$

The dual geometric program can be formed from the primal. Then using the same log transforms, a dual linear program can be obtained from the dual geometric program. Note that the geometric programs formed are a special type in that each constraint has one term or is a monomial.

## Condensation of Terms

Duffin (2) has shown that a geometric program can define a set of linear programs. This linearization is of use both analytically and computationally. An analytic example is described in (2) where the duality of geometric programming is proved using only linear programming theory. The computation methods of linear programs can thus be applied to geometric programs.

Duffin suggests a condensation of terms in a geometric program may be of computational use. The essential method of condensation is to take a constraint

$$g_k(x) = \sum_{i=1}^{T} u_i,$$

where

$$u_i = c_i x_1^{a_{i1}},\ldots,x_n^{a_{in}},$$

the $u_i$ being terms of the constraint.

Given a set of non-negative weights $\epsilon_1, \epsilon_2, \ldots, \epsilon_t$ such that

$$\sum_{i=1}^{T} \epsilon_i = 1,$$

then a condensed program or constraint can be obtained.

$$\bar{g}_k(x) = \prod_{i=1}^{T} \left(\frac{u_i}{\epsilon_i}\right)^{\epsilon_i},$$

thus

$$\bar{g}_k(x) = \bar{c}_k x_1^{\bar{a}_{k1}},$$

$$\bar{c}_k = \prod_{i=1}^{T} \left(\frac{c_i}{\epsilon_i}\right)^{\epsilon_i},$$

$$\bar{a}_{kj} = \sum_{i=1}^{T} \epsilon_i a_{ij}, \qquad j = 1, \ldots, N.$$

It is a direct consequence of the geometric inequality that

$$g_k(x) \geq \bar{g}_k(x).$$

This condensation of terms into monomials allows linear programming to be used as a computation method. This condensation can be used to reduce the number of terms in a problem with a resulting reduction in the degree of difficulty (number of terms - number of variables - 1).

A geometric program linearized using a set of weights is related to the original problem by

$$g_0(x) \geq \bar{g}_0(x).$$

Thus, a lower bound is obtained by solving the condensed problem. An efficient criterion for choosing initial weights and for adjusting weights at each iteration has not been developed.

A further development of the condensation principle allows geometric programs to handle negative coefficients. To illustrate, a geometric program must have constraints of the form

$$g_k(x) \leq 1,$$

where $g_k(x)$ is a posynomial. If a constraint $f(x)$ has negative coefficients, it can be written

$$f(x) = g_1(x) - g_2(x) \leq 1.$$

A new variable $x_{n+1} > 0$ can be added to form

$$g_1(x) \leq x_{n+1} \leq 1 + g_2(x).$$

Dividing by $x_{n+1}$ gives

$$\frac{g_1(x)}{x_{n+1}} \leq 1 \leq \frac{1 + g_2(x)}{x_{n+1}}.$$

This is equivalent to two constraints

$$G_1(x) = \frac{g_1(x)}{x_{n+1}} \leq 1,$$

$$G_2(x) = \frac{g_2(x) + 1}{x_{n+1}} \leq 1.$$

Let $\bar{G}_2(x)$ be a condensed posynomial $G_2(x)$. Now

$$\frac{1}{\bar{G}_2(\bar{x})} \leq 1$$

is a standard geometric programming constraint. A choice of weights, thus, will give a lower bound on the original problem.

The general method of condensation gives a lower bound on the original problem. Convergence to the optimal has not been demonstrated. Note condensation of problems with negative coefficients reduces the degree of difficulty by the number of terms condensed. This reduction can remove independence of primal variables if the degree of difficulty is negative. It should be noted there exist weights which make the resulting linear constraint set inconsistent.

## Constrained Maximization of Polynomials by Geometric Programming

Pascual and Ben-Israel (9) extended geometric programming to the maximization of a posynomial objective function subject to posynomial constraints.

This problem in a general form is

$$\text{minimize } (g_o(x))^{-1},$$

subject to

$$x_j > 0, \quad j = 1,\ldots,N,$$

$$g_i(x) \leq 1, \quad i = 1,\ldots,M.$$

Let $T_o$ be the number of terms in $g_o(x)$ and form a vector $\epsilon$ with $T_o$ components

$$\epsilon_k > 0, \quad k = 1,\ldots,T,$$

and

$$\sum_{k=1}^{T} \epsilon_k = 1.$$

Using the geometric inequality, a new objective function

$$\overline{g_o}(x,\epsilon) = \prod_{i=1}^{T} \left(\frac{c_i}{\epsilon_i}\right)^{\epsilon_i} \prod_{i=1}^{n} x_i^{\sum_{k=1}^{T} k^{a_{ki}}},$$

satisfies

$$g_o(x) \geq \overline{g}_o(x,\epsilon).$$

The above treatment of $g_o(x)$ is the same as Duffin's condensation (2). The problem is now in standard geometric programming form. By appending an index m to every term and exponent of the M + 1 polynomials, the dual can be written

$$\text{maximize } z(\epsilon_i,\delta) = \prod_{i=1}^{T} \left(\frac{c_{oi}}{\epsilon_i}\right)^{\epsilon_i} \prod_{m=1}^{M} \prod_{t=1}^{T_m} \left(\frac{c_{mt}\delta_{mo}}{\delta_{mt}}\right)^{\delta_{mt}},$$

where

$$\delta_{oo} = 1,$$

$$\lim_{\delta_{mt} \to 0} \frac{c_{mt}\delta_{mo}}{\delta_{mt}} = 1, \qquad n = 0,1,\ldots,M, \qquad t = 1,\ldots,T_m,$$

subject to the linear equality constraints

$$\sum_{t=1}^{T_o} \delta_{ot} = 1,$$

$$-\sum_{i=1}^{T} \epsilon_i a_{oin} + \sum_{m=o}^{M} \sum_{t=1}^{T_m} a_{mnt}\delta_{mt} = 0, \qquad n = 1,\ldots,N,$$

$$\delta_{mo} = \sum_{t=1}^{T_m} \delta_{mt}, \qquad m = 1,\ldots,M,$$

and non-negativity conditions

$$\delta_{mt} \geq 0, \qquad m = 0,1,\ldots,M, \qquad t = 0,1,\ldots,T_m.$$

It has been proven (9) that the minimum (maximum of $u(\epsilon,\delta)$ of the above problem), subject to the constraints on the vector $\epsilon$, solves the original problem. Also, the following relations hold at the optimum:

$$c_{oi} \prod_{i=1}^{N} x_i^{a_{oi}} = \frac{\epsilon_i}{z(\delta,\epsilon)}, \qquad i = 1,\ldots,T_o,$$

and

$$c_{mi} \prod_{i=1}^{N} x_i^{a_{mi}} = \frac{\delta_{mi}}{\delta_{mo}},$$

if

$$\delta_{mi} > 0, \qquad i = 1,\ldots,T_m, \qquad m = 1,2,\ldots,M.$$

The computation of the optimal using this method can be simplified for two special cases.

Case A: There is a unique solution $(\epsilon,\delta)$. This occurs when the exponent matrix $A = (a_{ij})$ is of rank m and if $N = M + 1$.

Case B: For any vector $\epsilon$, there is a unique solution to $\delta = \delta(\epsilon)$ of the dual constraints. This happens if $N - T_o = M =$ rank of A. In this case, minimize $\epsilon$ maximize $\delta$ $z(\epsilon,\delta)$ simplifies to minimize $\epsilon$ $z(\epsilon,\delta(\epsilon))$.

This computation method extends geometric programming to a larger class of problems. The idea of minimize-maximize is developed further in the next chapter.

## Decomposition of Geometric Programs

Heyman and Avriel (7) have developed a decomposition method for a special class of geometric programming problems. In many optimization problems, many variables appear in distinct groups with relative few variables appearing in more than one group. An example is the optimization of the goals of a multiplant firm. Only a few decision variables concern the total operation and most variables are restricted to decisions concerning only one plant.

A general form of the class of geometric programs that can be solved by decomposition is of the type

$$\text{minimize } P_o^o(z) + P_o^\iota(z,x^\iota),$$

subject to

$$P_k^o(z) \leq 1, \qquad k = 1,\ldots,p(o),$$

$$P_k(z,x^\iota) \leq 1, \qquad k = 1,\ldots,p(\iota),$$

$$z > 0, \qquad x^\iota > 0,$$

where z is a vector whose components are coupling variables with the main system and the $\iota$ subsystems. $P_o^o(z)$ is the posynomial that contains the objective function terms with primal variables contained in z. $P_o^\iota(z,x^\iota)$ is a posynomial that contains objective function terms that have coupling variables z

and variables $x^{\iota}$ of a distinct subsystem $\iota$.

$$P_o^o(z) \leq 1, \qquad k = 1,\ldots,p(o)$$

are constraints that have only coupling variables.

$$P_k^{\iota}(z,x^{\iota}) \leq 1, \qquad k = 1,\ldots,p(\iota),$$

are constraints that have coupling variables and variables of a distinct subsystem $\iota$.

The main results (7) of this decomposition method are:

Case A: Assume the number of coupling variables is equal to the total number of posynomial terms associated with the coupling subsystem. Assume for some subsystem the optimum of this geometric program dual has $\delta > 0$. Then the optimal solution vector $\delta^*$ of the original problem's dual can be expressed as a convex combination of the optimal dual subsystems.

Case B: Assume that the number of posynomial terms associated with the coupling system exceeds the number of components in the coupling vector by exactly one. The dual program of this coupling system has zero degrees of difficulty. The unique optimal is obtained from the coupling subsystem alone.

A more detailed explanation and proofs are contained in (7). A computation advantage of this method is that the resulting geometric programs are dimensionally smaller. The work required to determine the coupling system and the distinct

subsystems is a trade-off with less actual computation. In general, it should be noted that the system may not be decomposable.

# LINEARIZATION OF GEOMETRIC PROGRAMS

The development of another method of linearization of geometric programs is contained in this chapter. The justification for linearization is twofold; both the theory and computational methods of linear programming can be utilized. Linear programming theory (5) is the most developed and used of the mathematical programming methods. Proofs exist concerning computation convergence, duality relationships, existence of solutions, post-optimal analysis, and many other facets. Dantzig, as mentioned in Hadley (5, p. 30), is credited with developing the simplex method of linear programming. This computational algorithm has been further developed and used extensively since its general availability in 1951. Much literature on linear programming applications, theory, and computation is available and Hadley (5) is cited as only one of many references.

Linearization of geometric programs or generalized polynomial programs is achieved in several steps. A detailed explanation of each step is presented below. The general problem is expressed as

$$\text{maximize or minimize } g_o(x), \tag{3.1}$$

subject to

$$g_m(x) \leq 1, \qquad m = 1,2,\ldots,M,$$

where

$$g_m(x) = \sum_{i=1}^{T_m^I} u_{mi} - \sum_{i=T_{m+1}^I}^{T_m} u_{mi}, \tag{3.2}$$

where $T_m^I$ is the number of terms in $g_m(x)$ with positive coefficients and $T_m$ is total number of terms in $g_m(x)$. Where

$$u_{mi} = c_{mi} \prod_{j=1}^{N} x_j^{a_{mij}}, \tag{3.3}$$

with

$$0 < x_i < \infty, \quad i = 1,2,\ldots,N,$$

$$c_{mi} > 0, \quad i = 1,2,\ldots,T_m, \quad m = 0,1,\ldots,M,$$

$a_{mij}$ is an arbitrary real number.

Reduction of All $c_{mi}$ to Unity

A new constraint is added to above problem of the form

$$g_{M+1}(x) = x_{N+1}^{a_{M+1,1,N+1}} = C, \tag{3.4}$$

where C equals some $c_{mi}$ for which

$$c_{mi} \neq 1, \quad i = 1,2,\ldots,T_m, \quad m = 0,1,\ldots,M,$$

if $C > 1$, let

$$a_{M+1,1,N+1} = -1, \tag{3.5}$$

if $C < 1$, let

$$a_{M+1,1,N+1} = 1. \tag{3.6}$$

Now all $c_{mi}$ can be expressed as

$$c_{mi} = x_{N+1}^{a_{m,i,N+1}}, \tag{3.7}$$

where

$$a_{m,i,N+1} = \frac{\log(c_{mi})}{\log(C)}. \tag{3.8}$$

If this M + 1 constraint has strict equality at the optimum, the problems are equivalent.

## Linearization of Constraints

A method of linearization of posynomial constraints different from Duffin (2) will be described. Let a posynomial constraint be denoted by

$$g_k(x) = u_1 + u_2 + u_3 + \ldots + u_t \leq 1, \tag{3.9}$$

where

$$u_i = \prod_{j=1}^{N} x_j^{a_{ij}},$$

let a new variable $x_{N+1} > 0$ satisfy

$$u_1 + u_2 \leq x_{N+1} + u_3 + \ldots + u_t \leq 1. \tag{3.10}$$

Constraint equation (3.10) can be converted to

$$\frac{u_1}{x_{N+1}} + \frac{u_2}{x_{N+1}} \leq 1 \tag{3.11}$$

and

$$\bar{g}_k(x) = x_{N+1} + u_3 + \ldots + u_t \leq 1. \tag{3.12}$$

Let two weights $\epsilon_1$ and $\epsilon_2$ be selected such that

$$\epsilon_1 > 0, \quad \epsilon_2 > 0 \tag{3.13}$$

and

$$\epsilon_1 + \epsilon_2 = 1. \tag{3.14}$$

Now two constraints can be formed using $\epsilon_1$, $\epsilon_2$, and (3.11):

$$\frac{u_1}{x_{n+1}^{\epsilon_1}} \leq 1, \qquad \frac{u_2}{x_{n+1}^{\epsilon_2}} \leq 1. \tag{3.15}$$

For any selection of $\epsilon$ that satisfies (3.13) and (3.14), the constraints (3.15) and (3.12) are equivalent to (3.9).

The selection of new variables is repeated until $\bar{g}_k(x)$ has two terms. Each pair of constraints formed has an associated pair of weights $\epsilon$. When $\bar{g}_k(x)$ is reduced to two terms, it can be expressed as

$$x_{N+t-2} + u_t \leq 1. \tag{3.16}$$

A new pair of weights that satisfy (3.13) and (3.14) are used to form

$$\frac{x_{N+t-2}}{\epsilon_1} + \frac{u_t}{\epsilon_2} \leq 1. \tag{3.17}$$

Two term constraints require addition of one pair of weights. Constraints with more than two terms require T - 2 (T is number of terms) variables to be added and T - 1 pairs of weights.

## Linearization of Generalized Posynomials

A generalized polynomial constraint can be expressed as a combination of two posynomial functions:

$$g_m(x) - g_{m+1}(x) \leq 1, \tag{3.18}$$

adding $g_{m+1}(x)$ to both sides

$$g_m(x) \leq 1 + g_{m+1}(x). \tag{3.19}$$

Let P be defined as

$$P = K \prod_{i=1}^{N} x_i^{d_i}, \qquad K \text{ is a constant,} \tag{3.20}$$

where $d_i$ is selected such that no term of

$$\frac{g_m(x)}{P} \leq \frac{1}{P} + \frac{g_{m+1}(x)}{P} \tag{3.21}$$

is reduced to a constant. An additional restriction on P is given later. Select a variable $x_{n+1} > 0$ such that

$$\frac{g_m(x)}{P} \leq \frac{1}{P} + \frac{g_{m+1}(x)}{P} \leq x_{n+1}. \tag{3.22}$$

The equation (3.22) is used to form two constraints:

$$\frac{g_m(x)}{x_{n+1}P} \leq 1, \tag{3.23}$$

$$\frac{1}{x_{n+1}P} + \frac{g_{m+1}(x)}{x_{n+1}P} \leq 1. \tag{3.24}$$

If the restriction that original constraint (3.18) is active or tight at the optimal is added,

$$\frac{x_{n+1}}{\epsilon} \leq 1 \tag{3.25}$$

will have strict equality for some $0 < \epsilon_{n+1} < \infty$. The addition of variable $x_{n+1}$ and its associated constraints in general adds two terms to the system and the degree of difficulty is increased by one.

## Tight or Active Constraints

In general, it is not known what constraints are active or tight at the optimum of a geometric program prior to

solving the problem. Computation can be simplified if it is known all constraints are active at the optimal. This section describes a method that ensures all constraints are tight at the optimal. The most general constraint necessary to consider is

$$g_m(x) - g_{m+1}(x) \leq 1, \tag{3.26}$$

where $g_m(x)$ and $g_{m+1}(x)$ are posynomials. Select a new variable $x_{n+1} \geq 1$ that satisfies

$$x_{n+1}g_m(x) - g_{m+1}(x) = 1. \tag{3.27}$$

The variable $x_{m+1}$ can be restricted to satisfy

$$\frac{1}{\epsilon x_{m+1}} \leq 1, \tag{3.28}$$

where

$$0 < \epsilon \leq 1.$$

The two constraints (3.27) and (3.28) are used in place of the original (3.26). It can be seen that, in general, the addition of one variable and one additional constraint can be used to restrict any original constraint to being active or tight at the optimal.

### Conversion of an Objective Function to a Monomial

Let $g_o(x)$ be a geometric program's objective function and is positive at the optimum. Let

$$g_o(x) \leq \frac{1}{x_1}, \tag{3.29}$$

change the problem to

$$\text{minimize } \frac{1}{x_1}, \tag{3.30}$$

subject to additional constraint

$$g_o(x)x_1 \leq 1.$$

The problem of maximizing a polynomial is handled slightly different. Change problem to

$$\text{minimize } \frac{1}{x_1}, \tag{3.31}$$

subject to

$$x_1 \leq g_o(x). \tag{3.32}$$

If $g_o(x)$ is a monomial conversion, it can be completed by dividing both sides by $g_o(x)$. In general, $g_o(x)$ is not a monomial. Define a new constraint

$$\frac{x_1}{P} \leq \frac{g_o(x)}{P}, \tag{3.33}$$

where

$$P = K \prod_{i=2}^{N} x_i^{d_i}, \qquad K \text{ is a constant}, \tag{3.34}$$

where

$$d_i, \qquad i = 2,\ldots,N,$$

are selected such that no term of (3.33) is a constant. In a following section, linear equations are formed using the exponents of the variables in all constraints. The selection of P can be critical in forming a consistent system of linear equations.

A new variable $x_{n+1} > 0$ is selected such that

$$\frac{x_1}{P} \leq \frac{g_o(x)}{P} \leq x_{n+1}. \tag{3.35}$$

The inequality (3.35) is used to form two constraints:

$$\frac{x_1}{Px_{n+1}} \leq 1, \tag{3.36}$$

$$\frac{g_o(x)}{Px_{n+1}} \leq 1. \tag{3.37}$$

Strict equality is forced on (3.36) and (3.37) by adding a constraint

$$\frac{x_{n+1}}{\epsilon_{n+1}} \leq 1 \tag{3.38}$$

for some $0 < \epsilon_{n+1} < C$.

It can be seen that conversion of (3.32) to monomials is the same method required to convert a constraint of the form

$$1 \leq g_m(x),$$

which is not the normal form for geometric programming.

## Summary of Conversions

The preceding sections describe the conversions necessary to convert a mathematical program containing polynomial functions to a monomial constrained geometric program. Steps required are:

(a) Constraints that are not known to be active or tight at the optimal are forced to be tight by adding a variable to this constraint:

$$g_m(x) \leq 1,$$

adding new constraints

$$x_{n+1} g_m(x) = 1,$$

and

$$\frac{1}{\epsilon_{n+1} x_{n+1}} = 1,$$

where

$$0 < \epsilon_{n+1} \leq 1.$$

The degree of difficulty (number of terms - number of variables - 1) is unchanged.

(b) All terms are converted to unity coefficients

$$u_{mi} = c_i \prod_{j=1}^{N} x_j^{a_{mij}} = x_{N+1}^{a_{mi,N+1}} \prod_{j=1}^{N} x_j^{a_{mij}},$$

by the addition of one variable $x_{N+1} > 0$ and one constraint

$$g_{m+1}(x) = C x_{n+1}^{a_{m+1,1,n+1}} \leq 1, \qquad C \text{ is a constant.}$$

No change is made in degree of difficulty.

(c) Posynomial constraints of T terms

$$g_m(x) = \sum_{j=1}^{T} \prod_{i=1}^{N} x^{a_{mji}},$$

are converted to weighted constraint pairs.

$$\frac{1}{\epsilon_k} \prod_{i=1}^{N+1} x_i^{a_{mji}} \text{ or } \frac{1}{1-\epsilon_k} \prod_{i=1}^{N+1} x_i^{a_{m,j+1,i}},$$

where

$$0 < \epsilon_k < 1,$$

if T is greater than 2, t - 2 new variables are added. The total number of monomial constraints needed to convert a posynomial is (T - 1) x 2. Each constraint has one term. With (T - 1) x 2 new terms - T original terms gives a total of T - 2 additional terms, the degree of difficulty is unchanged.

(d) The objective function is converted to a constraint by the addition of one variable. This variable is then a one term objective function. The degree of difficulty is unchanged.

(e) Generalized polynomials

$$g_m(x) - g_{m+1}(x) \leq 1$$

are converted to two posynomial constraints by adding one variable and two terms. The degree of difficulty is increased by one.

## General Linear Form

A general form of the converted program can be written

minimize $\frac{1}{x_1}$,

subject to normal constraint

$$g_1(x) = Cx_2^{a_{12}} \leq 1,$$

subject to $T_1$ monomial converted constraints

$$g_m(x) = \prod_{i=1}^{N} x_i^{a_{mi}} \leq 1, \qquad m = 2,\ldots,M_1, \qquad T_1 = M_1 - 1,$$

subject to $T_2$ tightness constraints

$$g_m = \frac{1}{\epsilon_m} \prod_{i=1}^{N} x_i^{a_{mi}} \leq 1, \qquad m = M_1 + 1,\ldots,M_2,$$

$$T_2 = M_2 - M_1,$$

where

$$0 < \epsilon_m \leq 1,$$

subject to $T_3$ weighted pairs of constraints

$$g_m = \frac{1}{\epsilon_m} \prod_{i=1}^{N} x_i^{a_{mi}}, \qquad m = M_2 + 1,\ldots,M_3, \qquad T_3 = M_3 - M_2,$$

and

$$g_m = \frac{1}{1-\epsilon_{m-T_3}} \prod_{i=1}^{N} x_i^{a_{mi}}, \qquad m = M_3 + 1,\ldots,M_4,$$

$$T_3 = M_4 - M_3,$$

where

$$0 < \epsilon_m < 1,$$

subject to the possible $T_4$ tightness constraints needed to convert polynomials, reversed inequalities, or an objective function to minimization.

$$g_m(x) = \frac{x_m}{\epsilon_m} \leq 1, \qquad m = M_4 + 1,\ldots,M,$$

where

$$0 < \epsilon_m < \infty,$$

all variables satisfy $0 < x_n < \infty$ and all $a_{mi}$ are arbitrary real numbers.

The objective function of the dual geometric program is

$$\text{maximize } v(\delta) = (1)^{\delta_0}(c)^{\delta_1}\prod_{i=2}^{M_1}(1)^{\delta_i}\prod_{i=M_1+1}^{M_2}\left(\frac{1}{\epsilon_i}\right)^{\delta_1}$$

$$\prod_{k=M_2+1}^{M_3}\left(\frac{1}{\epsilon_k}\right)^{\delta_k}\prod_{l=M_3+1}^{M_4}\left(\frac{1}{\epsilon_l}\right)^{\delta_l}\prod_{r=M_4+1}^{M}\left(\frac{1}{\epsilon_r}\right)^{\delta_r}. \qquad (3.39)$$

In that the $\log(v(\delta))$ has the same maximizing point, the objective function simplifies to

$$\text{maximize } z = \sum_{i=M_1+1}^{M_2}\log\left(\frac{1}{\epsilon_1}\right)^{\delta_1} + \sum_{K=M_2+1}^{M_3}\log\left(\frac{1}{\epsilon_K}\right)^{\delta_K} + \sum_{L=M_3+1}^{M_4}$$

$$\log\left(\frac{1}{L}\right)^{L} + \sum_{r=M_4+1}^{M}\log\left(\frac{1}{r}\right)^{r}. \qquad (3.40)$$

The dual function is subject to linear constraints:

normality condition

$$\delta_0 = 1 \qquad (3.41)$$

orthogonality conditions

$$\prod_{j=0}^{M} a_{ji}^{\delta_j} = 0, \qquad i = 1,\ldots,N, \qquad (3.42)$$

$$\delta_i \geq 0.$$

Let (3.40), (3.41), and (3.42) be called linear program A.

It can be seen that given a vector $\epsilon$, the dual problem is a linear program. It is assumed in this linearization that (3.41) and (3.42) are consistent. The selection of P (3.20) and (3.34) is critical in forming (3.42). In that $\delta_0$ is unity requires at least one other $\delta_i$ to be positive. If any

row of (3.42) has $a_{ij}$ all of the same sign, only $\epsilon_i = 0$ for $a_{ij} \neq 0$ would satisfy the constraint. Usually a selection of P such that the $a_{ij}$'s are not of the same sign is sufficient for a consistent system. The necessary or sufficient conditions for selecting P have not been developed. The selection of P adds a constant to a subset of the $a_{ij}$ in a row. With the capability of adding a different constant to a subset of the elements in each row, formation of some consistent system should not be difficult in most cases.

A general outline of the computation necessary to solve the problem is:

(a) Solve the above linear program A for a given vector $\epsilon$.

(b) Use information from a basic feasible tableau of (a) to change the vector $\epsilon$.

(c) Use vector $\epsilon$ in (a) or terminate computation if the degree of significance achieved is satisfactory.

A geometric explanation of the computation method is to find a hyperplane defined by objective function (3.40) that is an optimal hyperplane defined by the linear constraints (3.41) and (3.42). Each feasible solution of the linear program defines an extreme of the hyperplane defined by (3.41) and (3.42). Each feasible solution to the linear program corresponds to N active or tight geometric program constraints. If all constraints are tight or active at the optimal, then all

extreme points must be optimal. A brief description of linear programming computation follows.

## Description of Linear Program Computation

The simplex method of solving the linear program A,

maximize $z = c'\delta$,

subject to

$$A\delta = b, \quad A \text{ is } n \times m, \quad \delta_i \geq 0, \quad b \geq 0,$$

can be described as a sequence of tableaus, the first of which is

| $C_B$ | Vectors in Basis | $x_B$ | $C_j$ Column Numbers |
|---|---|---|---|
| $C_{Bi}$ | | $x_{Bi}$ | $y_{ij}$ |
| | $z_j - C_j$ row | $z$ | $z_j - C_j$ |

Where

$$y_{ij} = a_{ij}, \quad i = 1,2,\ldots,N, \quad j = 1,2,\ldots,M,$$

an identity matrix forms the last $N$ columns of $y_{ij}$. $C_B$ is the vector of cost coefficients of variables in the basis. $x_B$ is the vector of variables in the basis. $C_j$ is the cost

coefficient of the $x_j$ variable. z is the computed value of the objective function using $C_B^t x_B$:

$$z_j - c_j = \sum_{i=1}^{N} c_{Bi} y_{ij}.$$

The initial tableau has artificial variables j = M + 1,..., M + N as basic variables $x_B$. Columns $C_B$ contains the corresponding cost coefficients. These artificial variables are given initial values of b and thus form a basic feasible solution. The components of $C_B$ are assigned large negative values such that, at the optimal, the artificial variables have no effect on the objective function value. Each iteration generates a new tableau by substitution of a column vector not in basis for a column vector in the basis.

The column to enter basis k is selected such that k = j, j corresponds to

$$\underset{j}{\text{minimize}}\ (z_j - c_j), \quad \text{with } (z_1 - c_j) < 0, \quad j = 1,\ldots,M.$$

The variable and associated column to be removed from the basis is $x_{Br}$, r = i such that

$$\underset{i}{\text{minimize}}\ \left(\frac{x_{Bi}}{y_{ri}}\right), \quad y_{ri} \geq 0, \quad i = 1,2,\ldots,N.$$

A new tableau is generated using the pivot element $y_{rk}$ with the following relations:

$$\text{new} \quad y_{ij} = y_{ij} - \frac{y_{ik}}{y_{rk}} y_{rj}, \quad \text{all } j, \quad i = 1,\ldots,N + 1, \quad i \neq r,$$

$$\text{new} \quad y_{rj} = \frac{y_{rj}}{y_{rk}}, \qquad \text{all } j,$$

where

$$x_B = y_{o}, \qquad z = y_{N+1,o},$$

$$z_j - c_j = y_{N+1,j}, \qquad j = 1,\ldots,M.$$

Computation is terminated when all $z_j - c_j \geq 0$. The linearized geometric program is solved by this simplex method.

## Optimal Analysis

The termination of the simplex method is the condition

$$z_j - c_j \geq 0, \qquad \text{all } j.$$

The $z_j - c_j$ associated with the basis are equal to zero. Each column of the linear program is associated with a monomial constraint of the linearized geometric program. All columns that have $z_j - c_j$ equal to zero correspond to active or tight constraints. When given any initial vector $\epsilon$ used to linearize a geometric program, only a subset of the constraints must be active or tight at the optimal linear program solution. The optimum of the linear program is thus, in general, not an optimum of the original problem. The weights $\epsilon$ can be adjusted so as to reduce the linear program optimal. Only the cost coefficients of the linear program are a function of the weights assigned. The optimal of linear program A is

$$z = c_B^j \delta_b$$

where $C_B$ is the cost coefficient of the variables $\epsilon_B$ in the optimal basis. Let $\Delta E_j$ be a small change in $\epsilon_j$. Then the objective function's dependence on $\epsilon_i$ and $\Delta E_i$ can be expressed as

$$\Delta z = \sum_{i=1}^{N} \frac{\partial C_{Bi}}{\partial \epsilon_{Bi}} \Delta E_{Bi} \delta_{bi}.$$

Likewise, the changes in $z_j - c_j$ row can be expressed as

$$\Delta(z_j - c_j) = \sum_{i=1}^{N} \frac{\partial C_{Bi}}{\partial \epsilon_{Bi}} \Delta E_{Bi} y_{ij} - \frac{\partial C_j}{\partial \epsilon_j},$$

$j = 1,\ldots,M,$ where $j$ is not in basis of A.

For the linear program to be optimal, the $z_j - c_j + \Delta(z_j - c_j)$ must be non-negative. A new linear program B can be formulated to change the weights:

$$\text{maximize} - \sum_{i=1}^{N} \frac{\partial C_{Bi}}{\partial \epsilon_{Bi}} \delta_{bi} \Delta E_{Bi},$$

subject to linear constraints

$$\sum_{i=1}^{N} \frac{\partial C_{Bi}}{\partial \epsilon_{Bi}} y_{ij} \Delta E_{Bi} - \frac{\partial C_j}{\partial \epsilon_j} + z_j - c_j = 0,$$

$j = 1,\ldots,M,$ where $j$ is not in basis of A,

where

$$C_{Bi} = \log \frac{1}{\epsilon_{Bi}},$$

$$\frac{\partial C_{Bi}}{\partial \epsilon_{Bi}} = \frac{-1}{\epsilon_{Bi}},$$

$$\delta_{Bi} \geq 0,$$

$\Delta E_j$ unrestricted in sign.

Let this linear program be denoted as linear program B.

It can be noted that the number of constraints is equal to the degree of difficulty of the transformed problem. In that $\triangle E_j$ is unrestricted, an additional column is added for each $\triangle E_j$ which allows the standard simplex algorithm to be used.

This linear program has as its solution a feasible direction for decreasing z. The optimal set of $\triangle E_j$ is scaled down such that $\epsilon_j + \triangle E_j$ satisfies the appropriate constraint on $\epsilon_j$. The new weights $(\epsilon_j + \triangle E_j)$ are substituted for $\epsilon_j$ in linear program A. The z and $z_j - c_j$'s of the first linear program are computed and the simplex method starts from the existing tableau.

If the computed sum of squares of $z_j - c_j$ is not lower than the previous sum of squares of $z_j - c_j$, the vector $\triangle E$ is divided by two until a reduction is obtained. When the sum of squares of $z_j - c_j$ is reduced, a new linear program B is solved to obtain another feasible direction.

The algorithm is terminated when the sum of squares of the vector $z_j - c_j$ is below some predetermined limit. It can be noted that linear program A may be unbounded for a set of initial weights. No problem is encountered, though, if the reduction of the norm of $z_j - c_j$ is used. The criterion of reducing $z_j - c_j$ also allows the same basic feasible tableau of linear program A to be used for all calculations.

## Post Optimal Analysis

The objective function of the linear program has cost coefficients of three types:

(a) $\log \frac{1}{\epsilon_j}$,

where $0 < \epsilon_j < \infty$,

(b) $\log \frac{1}{1 - \epsilon_K}$,

where $0 < \epsilon_K < 1$,

(c) constant C or 0.

Cost coefficients of the form (a) and (b) are convex and differentiable in the defined region. Each $\epsilon_j$ of the objective function is non-negative. The objective function thus is a summation of convex functions in $\epsilon$ multiplied by a non-negative scalar. Thus, from the theory of convex programming (13), it follows there exists a vector $\epsilon$ which would provide a global minimum of the objective function. However, this does not imply global convergence for any algorithm based on iteration of vector $\epsilon$.

Each of the geometric program constraints is of form

$$g_m(x) = \left(\frac{1}{m}\right) \prod_{i=1}^{N} x^{a_{mi}} \leq 1. \tag{3.43}$$

Given $\epsilon_m$ and the fact that strict equality holds at the optimal, inequality (3.43) can be written

$$\prod_{i=1}^{N} x^{a_{mi}} = \epsilon_m. \tag{3.44}$$

Taking the logarithm of both sides gives

$$\sum_{i=1}^{N} e_{mi} \log x_i = \log \epsilon_m. \qquad (3.45)$$

Given N constraints that form an independent system of equations of the form (3.45), the logarithm of the primal variables can be obtained.

Computation, though, is simplified by using the final linear program A's tableau. The columns associated with the initial identity matrix contain the inverse of those columns in the final basis. The rank of this set of columns is N. Note the left-hand side of (3.45) is a column of the original linear program. The inverse of the N equations of form (3.45) is the transpose of the columns containing the original identity matrix. One matrix multiplication gives the logarithms of all primal variables.

## Constrained Maximization of Posynomials

A computer program (Appendix) was written using the above conversions. This section and the next section contain examples of non-standard geometric programs. The example of this section is from Pascual and Ben-Israel (9):

$$\text{maximize } x_1 + x_2, \tag{3.46}$$

subject to

$$x_1 > 0, \quad x_2 > 0,$$

$$x_1^2 + x_2^2 \leq 1, \tag{3.47}$$

$$\frac{2x_1}{5} + \frac{x_2}{5} \leq 1, \tag{3.48}$$

$$2x_1^{-1}x_2 \leq 1. \tag{3.49}$$

It is known that all of the constraints are tight at the optimal; thus, tightness constraints are not needed. A normal variable $x_3 > 0$ is added and a new constraint

$$\frac{x_3}{2} = 1. \tag{3.50}$$

Using the fact that $x_3 = 2$, (3.48) becomes

$$x_1 x_3^{-.161} + x_2 x_3^{-1.161} \leq 1, \tag{3.51}$$

inequality (3.49) becomes

$$x_1^{-1} x_2 x_3 \leq 1. \tag{3.52}$$

This completes normalization of the coefficients.

The new objective function is

$$\text{minimize } \frac{1}{x_4},$$

subject to

$$x_4 \leq x_1 + x_2. \tag{3.53}$$

Dividing both sides of (3.53) by $x_1 x_2$ gives

$$\frac{x_4}{x_1 x_2} \leq \frac{1}{x_2} + \frac{1}{x_1}. \tag{3.54}$$

Select $x_5 > 0$ that satisfies

$$\frac{x_4}{x_1 x_2} \leq \frac{1}{x_2} + \frac{1}{x_1} \leq x_5. \tag{3.55}$$

Inequality (3.55) is used to form

$$\frac{x_4}{x_1 x_2 x_5} \leq 1, \tag{3.56}$$

$$\frac{1}{x_2 x_5} + \frac{1}{x_1 x_5} \leq 1. \tag{3.57}$$

Inequality (3.57) is used to form two constraints:

$$\frac{1}{\epsilon_1 x_2 x_5} \leq 1, \qquad \frac{1}{(1-\epsilon_1)x_1 x_5} \leq 1, \tag{3.58}$$

where $0 < \epsilon_1 < 1$. A tightness constraint of

$$\frac{x_5}{\epsilon_2} \leq 1, \tag{3.59}$$

where $0 < \epsilon_2 < \infty$, is added to bound $x_5$. Some $\epsilon_2$ forces tightness.

Inequality (3.47) is used to form two monomial constraints:

$$\frac{x_1^2}{\epsilon_3} \leq 1, \qquad \frac{x_2^2}{1 - \epsilon_3} \leq 1, \tag{3.60}$$

where $0 < \epsilon_3 < 1$.

Inequality (3.51) is used to form two monomial constraints

$$\frac{x_1 x_3^{-.161}}{\epsilon_4} \leq 1, \qquad \frac{x_2 x_3^{-1.161}}{1 - \epsilon_4} \leq 1. \tag{3.61}$$

A new statement of the problem using the monomial constraints is

$$\text{minimize } g_0(x) = \frac{1}{x_4},$$

subject to

$$g_1(x)\ \frac{x_3}{2} \leq 1,$$

$$g_2(x)\ \frac{x_4}{x_1 x_2 x_5} \leq 1,$$

$$g_3(x) = \frac{1}{\epsilon_1 x_2 x_5} \leq 1, \qquad g_4(x) \frac{1}{1 - \epsilon_1} \frac{1}{x_1 x_5} \leq 1,$$

$$g_5(x) \frac{x_5}{\epsilon_2} \leq 1,$$

$$g_6(x) \frac{x_1^2}{\epsilon_3} - 1, \qquad g_7(x) \frac{x_2^2}{1 - \epsilon_3} \leq 1,$$

$$g_8(x) \frac{x_1 x_3^{-.161}}{\epsilon_4} \leq 1, \qquad g_9(x) \frac{x_2 x_3^{-1.161}}{1 - \epsilon_4} \leq 1,$$

$$g_{10}(x) = x_1^{-1} x_2 x_3 \leq 1,$$

where

$$0 < \epsilon_1 < 1, \qquad 0 < x_i \quad , \quad i = 1,\ldots,5,$$

$$0 < \epsilon_2 < 10,$$

$$0 < \epsilon_3 < 1,$$

$$0 < \epsilon_4 < 1.$$

The dual geometric program is

$$\text{maximize } \log \frac{1}{2} \delta_1 + \log \frac{1}{\epsilon_1} \delta_3 + \log \frac{1}{1 - \epsilon_1} \delta_4 + \log \frac{1}{\epsilon_2} \delta_5$$

$$+ \log \frac{1}{\epsilon_3} \delta_6 + \log \frac{1}{1 - \epsilon_1} \delta_7 + \log \frac{1}{\epsilon_4} \delta_8 +$$

$$\log \frac{1}{1 - \epsilon_4} \delta_9,$$

subject to linear constraints

$$\delta_0 = 1,$$

$$(x_1) \quad \delta_2 - \delta_4 + 2\delta_6 + \delta_8 - \delta_{10} = 0,$$

$$(x_2) \quad -\delta_2 - \delta_3 + 2\delta_7 + \delta_8 + \delta_{10} = 0,$$

$$(x_3) \quad +\delta_1 - .161\delta_8 - 1.161\delta_9 + \delta_{10} = 0,$$

$$(x_4) \quad -\delta_0 + \delta_2,$$

$$(x_5) \quad -\delta_2 - \delta_3 - \delta_4 + \delta_5 = 0,$$

$$\delta_i \geq 0, \qquad i = 0,\ldots,10,$$

let $\epsilon_1,\epsilon_2,\epsilon_3 = .5$, let $\epsilon_4 = 1.0$.

A measure of the convergence of the computation method is given below. Each iteration required a linear program computation with five constraints. The number of constraints is equal to the degree of difficulty of the original problem.

| Iteration Number | Computed z | Sum of Squares of $z_j - c_j$ |
|---|---|---|
| Initial | 7.2262 | 667.30 |
| 1 | 2.4551 | 94.122 |
| 2 | .19398 | 2.9933 |
| 3 | - .2744 | .058108 |
| 4 | - .29384 | $.2472 \times 10^{-7}$ |
| 5 | - .29389 | $.4455 \times 10^{-18}$ |

At the completion of iteration 5, the values of $z_j - c_j$ are:

| | $z_j - c_j$ | |
|---|---|---|
| $j = 1$ | 0.0 | basic, |
| $j = 2$ | 0.0 | basic, |
| $j = 3$ | $-.2235 \times 10^{-9}$, | |
| $j = 4$ | 0.0 | basic, |
| $j = 5$ | 0.0 | basic, |
| $j = 6$ | $-.7195 \times 10^{-10}$, | |
| $j = 7$ | $.3750 \times 10^{-9}$, | |
| $j = 8$ | 0.0 | basic, |
| $j = 9$ | $.4470 \times 10^{-9}$. | |

The computed vector at iteration 5 is

$$\epsilon_1 = .6666666,$$

$$\epsilon_2 = .3354101,$$

$$\epsilon_3 = .8000000,$$

$$\epsilon_4 = .8000000.$$

The value of the objective function at iteration 5 is -.29389 which is the log $\frac{1}{x_4}$. The original problem's objective function must equal $\frac{1}{x_4}$, thus

$$\frac{1}{x_4} = e^{-.29389} = \frac{\sqrt{5}}{3}.$$

The original variables $x_1$ and $x_2$ can be obtained from constraints $g_6(x)$ and $g_7(x)$ given $G_3$:

$$g_6(x) = \frac{x_1^2}{\epsilon_3} = 1, \qquad x_1 = \sqrt{.8} = \frac{2}{\sqrt{5}}$$

$$g_7(x) = \frac{x_2^2}{1 - \epsilon_3} = 1, \qquad x_2 = \sqrt{.2} = \frac{1}{\sqrt{5}}.$$

These answers correspond to those given in (9).

## Generalized Polynomial Programs

The example in this section is taken from Blau (1, p. 125). The problem is to minimize $g_0(x)$ with prior knowledge that $g_0(x) < 0$ and that all constraints are tight at the optimal. The problem is changed to maximize $-g_0(x)$ so that the computed optimal is positive. A statement of the problem is

$$\text{maximize } 50x_1x_2 + 30x_1^2 - 5x_1^3 - 10x_2^2,$$

subject to

$$x_1^2 + x_2^2 \geq 1{,}200, \tag{3.62}$$

$$x_1 > 0, \qquad x_2 > 0. \tag{3.63}$$

The addition of a normal constraint

$$\frac{1}{5x_3} \leq 1 \tag{3.64}$$

allows the problem to be converted to

$$\text{maximize } x_1x_2x_3^{2.43} + x_1^2x_3^{2.11} - x_1^3x_3 - x_2^2x_3^{1.43}, \tag{3.65}$$

subject to

$$x_1^2 + x_2^2 \geq x_3^{4.40}, \tag{3.66}$$

$$x_1 > 0, \qquad x_2 > 0, \qquad x_3 = .2. \tag{3.67}$$

The objective function is converted to a monomial by selecting $x_4 > 0$ such that it satisfies

$$\text{minimize } \frac{1}{x_4} \tag{3.68}$$

subject to

$$x_4 \geq x_1 x_2 x_3^{2.43} + x_1^2 x_3^{2.11} + x_1^3 x_3 + x_2^2 x_3^{1.43}. \tag{3.69}$$

The addition of the last two terms to both sides gives

$$x_1^3 x_3 + x_2^2 x_3^{1.43} + x_4 \geq x_1 x_2 x_3^{2.43} + x_1^2 x_3^{2.11}. \tag{3.70}$$

Dividing both sides by $x_1^{.5} x_2^{.5} x_3^{4.29}$ and selecting a new variable $x_5 > 0$ gives

$$\frac{x_1^{2.5}}{x_2^{.5} x_3^{3.29}} + \frac{x_2^{1.5}}{x_1^{.5} x_3^{2.86}} + \frac{x_4}{x_1^{.5} x_2^{.5} x_3^{4.29}} \geq \frac{x_1^{.5} x_2^{.5}}{x_3^{1.86}} + \frac{x_1^{1.5}}{x_2^{.5} x_3^{2.17}} \geq x_5. \tag{3.71}$$

Inequality (3.71) can be written as two inequalities:

$$\frac{x_1^{2.5}}{x_2^{.5} x_3^{3.29} x_5} + \frac{x_2^{1.5}}{x_1^{.5} x_3^{2.86} x_5} + \frac{x_4}{x_1^{.5} x_2^{.5} x_3^{4.29} x_5} \geq 1 \tag{3.72}$$

and

$$\frac{x_1^{.5} x_2^{.5}}{x_3^{1.86} x_5} + \frac{x_1^{1.5}}{x_2^{.5} x_3^{2.17}} \geq 1. \tag{3.73}$$

Select a new variable $x_6 > 0$ that satisfies

$$\frac{x_1^{2.5}}{x_2^{.5} x_3^{3.29} x_5} + \frac{x_2^{1.5}}{x_1^{.5} x_3^{2.86} x_5} \geq x_6 + \frac{x_4}{x_1^{.5} x_2^{.5} x_3^{4.29} x_5} \geq 1. \tag{3.74}$$

The left two terms are converted to monomials

$$\frac{x_1^{2.5}}{\epsilon_1 x_2^{.5} x_3^{3.29} x_5 x_6} \leq 1 \quad \text{and} \quad \frac{x_2^{1.5}}{(1-\epsilon_1) x_1^{.5} x_3^{2.86} x_5 x_6}, \tag{3.75}$$

where $0 < \epsilon_1 < 1$. The remaining part of (3.74) is converted to monomials

$$\frac{x_6}{\epsilon_2} \leq 1 \quad \text{and} \quad \frac{x_4}{(1-\epsilon_2) x_1^{.5} x_2^{.5} x_3^{4.29} x_5} \leq 1, \tag{3.76}$$

where $\epsilon_2$ satisfies $0 < \epsilon_2 < 1$. Inequality (3.73) is converted to monomials

$$\frac{x_1^{.5} x_2^{.5}}{\epsilon_3 x_3^{1.86} x_5} \leq 1 \quad \text{and} \quad \frac{x_1^{1.5}}{(1-\epsilon_3) x_2^{.5} x_3^{2.17}} \leq 1, \tag{3.77}$$

where $0 < \epsilon_3 < 1$. A tightness constraint is added of the form

$$\frac{x_5}{\epsilon_4} \leq 1, \tag{3.78}$$

where $\epsilon_4 > 0$.

Inequality (3.66) is rewritten as

$$x_3^{4.40} \leq x_1^2 + x_2^2. \tag{3.79}$$

Dividing by $x_3^{4.29}$ and selecting a new variable $x_7 > 0$ gives

$$\frac{x_3^{.113}}{x_7} \leq \frac{x_1^2}{x_3^{4.29} x_7} + \frac{x_2^2}{x_3^{4.29} x_7} \leq 1. \tag{3.80}$$

The left-hand side forms a monomial constraint

$$\frac{x_3^{.113}}{x_7} \leq 1. \tag{3.81}$$

The remaining inequality is converted to two monomial constraints

$$\frac{x_1^2}{\epsilon_5 x_3^{4.29} x_7} \leq 1 \quad \text{and} \quad \frac{x_2^2}{(1-\epsilon_5) x_4^{4.29} x_7} \leq 1, \tag{3.82}$$

where $\epsilon_5$ satisfies $0 < \epsilon_5 < 1$. A tightness constraint is added:

$$\frac{x_7}{\epsilon_6} \leq 1,$$

where $\epsilon_6 > 0$.

The use of the scale factor $x_3^{4.29} = 1{,}000$ when converting the problem constraints decreases the number of iterations required to solve the problem. The scale factor is not necessary, but computational experience indicates that a rough estimate of the optimal solution can be obtained in a few iterations irrespective of the magnitude of the vector $\epsilon$'s components. If the components of $\epsilon$ are significantly different than one, convergence is slow once a close approximation is obtained. Scaling of constraints after determining a rough estimate $\left(\sum_{j=1}^{M} (z_j - c_j)^2\right) < \text{limit}$ could be used to improve convergence.

The resulting converted geometric program is

$$\text{minimize } g_0(x) = \frac{1}{x_4},$$

subject to

$$g_1(x) = \frac{x_3}{5} = 1,$$

$$g_2(x) = \frac{x_1^{2.5}}{\epsilon_1 x_2^{.5} x_3^{3.29} x_5 x_6} \leq 1, \qquad g_3(x) = \frac{x_2^{1.5}}{(1-\epsilon_1) x_1^{.5} x_3^{2.86} x_5 x_6},$$

$$g_4(x) = \frac{x_6}{\epsilon_2} \leq 1, \qquad g_5(x) = \frac{x_4}{(1-\epsilon_2) x_1^{.5} x_2^{.5} x_3^{4.29} x_5} \leq 1,$$

$$g_6(x) = \frac{x_1^{.5} x_2^{.5}}{\epsilon_3 x_3^{1.86} x_5} \leq 1, \qquad g_7(x) = \frac{x_1^{1.5}}{(1-\epsilon_3) x_2^{.5} x_3^{2.17}} \leq 1,$$

$$g_8(x) = \frac{x_5}{\epsilon_4} \leq 1,$$

$$g_9(x) = \frac{x_3^{.113}}{x_7} \leq 1,$$

$$g_{10}(x) = \frac{x_1^{2}}{\epsilon_5 x_3^{4.29} x_7} \leq 1, \qquad g_{11}(x) = \frac{x_2^{2}}{(1-\epsilon_5) x_4^{4.29} x_7} \leq 1,$$

$$g_{12}(x) = \frac{x_7}{\epsilon_6} \leq 1,$$

where

$$0 < \epsilon_1 < 1,$$

$$0 < \epsilon_2 < 1,$$

$$0 < \epsilon_3 < 1,$$

$$0 < \epsilon_4 < \infty,$$

$$0 < \epsilon_5 < 1,$$

$$0 < \epsilon_6 < \infty,$$

and

$$x_i > 0, \quad i = 1,2,\ldots,7.$$

The dual geometric program converts to

$$\begin{aligned}\text{maximize } & \log \tfrac{1}{5}\,\delta_1 + \log \tfrac{1}{\epsilon_1}\,\delta_2 + \log \tfrac{1}{1-\epsilon_1}\,\delta_3 + \log \tfrac{1}{\epsilon_2}\,\delta_4 \\ & + \log \tfrac{1}{1-\epsilon_2}\,\delta_5 + \log \tfrac{1}{\epsilon_3}\,\delta_6 + \log \tfrac{1}{1-\epsilon_3}\,\delta_7 \\ & + \log \tfrac{1}{\epsilon_4}\,\delta_8 + \log \tfrac{1}{\epsilon_5}\,\delta_{10} + \log \tfrac{1}{1-\epsilon_5}\,\delta_{11} \\ & + \log \tfrac{1}{\epsilon_6}\,\delta_{12},\end{aligned}$$

subject to

$$\delta_0 = 1,$$

$$(x_1) \quad 2.5\delta_2 - .5\delta_3 - .5\delta_5 + .5\delta_6 + 1.5\delta_7 + 2\delta_{10} = 0,$$

$$(x_2) \quad -.5\delta_2 + 1.5\delta_3 - .5\delta_5 + .5\delta_6 + 1.5\delta_7 + 2\delta_{11} = 0,$$

$$(x_3) \quad \delta_1 - 3.29\delta_2 - 2.86\delta_3 - 4.29\delta_5 - 1.86\delta_6 - 2.17\delta_7 + .113\delta_9 - 4.29\delta_{10} - 4.29\delta_{11} = 0,$$

$$(x_4) \quad -\delta_0 + \delta_5 = 0,$$

$$(x_5) \quad -\delta_1 - \delta_2 - \delta_5 - \delta_6 - \delta_7 + \delta_8 = 0,$$

$$(x_6) \quad -\delta_2 - \delta_3 + \delta_4 = 0,$$

$$(x_7) \quad -\delta_9 - \delta_{10} - \delta_{11} + \delta_{12} = 0,$$

$$\delta_i \geq 0, \quad i = 1,\ldots,12,$$

let $\epsilon_1,\epsilon_2,\epsilon_3,\epsilon_5 = .5$ and $\epsilon_4,\epsilon_6 = 1$.

A measure of the convergence of the method is given below:

| Iteration Number | Computed z | Sum of Squares of $z_j - c_j$ |
|---|---|---|
| Initial | 1.732 | 6.824 |
| 1 | .407 | 6.463 |
| 2 | -1.301 | 5.409 |
| 3 | -3.213 | 4.243 |
| 4 | -6.358 | 3.217 |
| 5 | -8.440 | .602 |
| 6 | -8.414 | .0457 |
| 7 | -8.464 | .00158 |
| 8 | -8.4492 | $.9388 \times 10^{-6}$ |
| 9 | -8.4492 | $.308 \times 10^{-12}$ |

At the completion of iteration 9, the values of $z_j - c_j$ are:

| | $z_j - c_j$ | |
|---|---|---|
| $j = 1$ | $0.898 \times 10^{-17}$, | |
| $j = 2$ | $0.126 \times 10^{-8}$, | |
| $j = 3$ | 0.0 | basic, |
| $j = 4$ | 0.0 | basic, |
| $j = 5$ | 0.0 | basic, |
| $j = 6$ | $0.744 \times 10^{-8}$, | |
| $j = 7$ | 0.0 | basic, |

$j = 8$ 0.0 basic,

$j = 9$ $-0.555 \times 10^{-6}$,

$j = 10$ $0.508 \times 10^{-8}$,

$j = 11$ 0.0 basic,

$j = 12$ 0.0 basic.

The computed vector $\epsilon$ at iteration 9 is

$\epsilon_1 = .47643,$

$\epsilon_2 = .81054,$

$\epsilon_3 = .81306,$

$\epsilon_4 = 1.23137,$

$\epsilon_5 = .12803,$

$\epsilon_6 = 1.20000,$

The value of the linear program at iteration 9 is -8.4492. This corresponds to the $\log \frac{1}{x_4}$. In that $x_4$ corresponds to the negative of the original problem's objective function, it is

$$\text{optimal} = -e^{8.4492} = -4672.$$

The value of $x_1$ and $x_2$ can be obtained from the converted constraints $g_{10}(x)$ and $g_{11}(x)$. In that

$$x_1^2 = 1{,}200 \times \epsilon_5$$

and

$$x_2^2 = 1{,}200 + (1 - \epsilon_5),$$

the computed values are

$x_1 = 12.45,$ $x_2 = 33.2.$

The optimal values as given in Blau (1) are

$z = -4677.5678,$

$x_1 = 12.583332,$

$x_2 = 32.274754.$

# CONCLUSIONS AND RESULTS

The usefulness of the linearization of geometric programs can be measured relative to other methods of optimization. Geometric programming is restricted to posynomial functions with all variables strictly positive. Generalized polynomial programs allow all polynomials but also require the variables to be strictly positive. The SUMT and Lagrange multiplier method do not have either restriction.

The significant difference in handling constraints with SUMT and linearization make computational comparisons highly problem dependent. A rough comparison can be made among the Lagrange method, Blau's algorithm, and linearization. Blau's algorithm handles generalized polynomial programs with T terms, N variables, and M constraints. It is required that all constraints are active or tight at the optimal and that the signs of the constraints at the optimal are known. With the problem so restricted, the Lagrange multiplier method requires inversion of a $(T + N + 2M + 1)$ matrix at each iteration. Blau's algorithm reduces the size of the matrix to be inverted at each iteration to $(N + M + 1)$. The linearization described in this paper requires a linear program to be solved at each iteration. The number of constraints of this linear program correspond to the degree of difficulty of the linearized geometric program (number of terms - number of variables - 1). The linearization of a geometric program does not change the

degree of difficulty. When a generalized polynomial program is linearized, the degree of difficulty is increased one for each constraint that contains a negative coefficient or a reversed inequality. Thus, a linearized program has a degree of difficulty that ranges from (T - N) to (T + M - N). Hadley (6, p. 134) states the number of simplex iterations required to solve a linear program ranges from the number of constraints to two times the number of constraints. A rough computational estimate of linear programs is 1.5 x (computation required to invert a matrix of order equal to number of constraints of the linear program).

For comparison purposes, a computation estimate based on the size of matrix to be inverted at each iteration is:

| Lagrange multiplier | Blau's algorithm | Linearized method |
|---|---|---|
| T + N + 2M + 1 | N + M + 1 | 1.5(T - N) to 1.5(T + M - N) |

where T = number of terms in original problem

N = number of variables in original problem

M = number of constraints in original problem

The linearization clearly requires less computation for each iteration than the Lagranian method. Computation

comparisons of the linearized method and Blau's algorithm are problem dependent.

The computation method of solving the preceding linearized geometric program can be classed as a method of feasible directions. Zoutendijk (13) treats the standard simplex method as only one method of feasible directions. A more complete computational comparison of the linearization of geometric programs and other methods could be made using the analysis of Zoutendijk (13).

The linearized method does not guarantee in general cases a convergence to a global optimal. However, since the algorithm uses parametric simplex routines, post optimal analysis of sensitivity and related tests (13) can be applied to test for global convergence.

An advantage of the linearization method in geometric programming is that primal variables can be determined using the optimal weighting vector . Geometric programs, the number of possible equations available to determine the primal variables, are restricted to the number of terms in tight constraints. This in general is not sufficient to determine all primal variables.

The present linearization development removes the requirement that all constraints be tight at the optimal for generalized polynomial programs. The introduction of tightness variables and constraints also means that the signs of the constraints need not be known before solving the problem.

The sign of the objective function at the optimal, though, is required in the present linearization.

The sensitivity of constraints and variables on the objective function of a linearized geometric program have not been developed. Computational experience with the second example in Chapter 3 indicates the present algorithm's main use may be to obtain an approximation, which may be used as an initial point for another method. The sensitivity of the components of the vector $\epsilon$ on the objective function may be a starting point for further developments in the area of stochastic geometric programming.

Further developments of the linearization method are needed. The required conversions to linearize a general polynomial program are tedious and error prone. The development of a computer program to automate this process is necessary for the linearization method to have practical use. It should be noted that the program in the appendix does not have this capability. The program assumes a consistent linear constraint set as input data.

The selection of $P$ in the sections that dealt with generalized polynomials, reversed inequalities, or conversion of maximization problems to minimization problems requires further development. The entire linearization method is based on the existence of a consistent linear program. The formation of this consistent program is not unique. It can be noted that Blau (1) forms a linear constraint set by

using the exponents of the reciprocals of terms corresponding to negative coefficients. Pascual and Ben-Israel (9) and Duffin (2) used the exponents of the reciprocal of a weighted set of terms. Neither of these methods of forming a consistent linear constraint set are in general sufficient or necessary. The development of the necessary and sufficient conditions, if possible, would allow a large class of non-linear systems and unconstrained optimization problems to be solved without the problem of converging to stationary points or local optimums. A possible modification of the simplex method may be used to select $P_i$ to form consistent linear programs dynamically. In conclusion, it appears more and larger problems were uncovered than solved.

# LITERATURE CITED


1. Blau, Gary E. Generalized polynomial programming: extensions and applications. Unpublished Ph.D. thesis. Stanford, California, Library, Stanford University. 1968.

2. Duffin, R. J. Linearizing geometric programs. SIAM. Review 12:211-227. 1970.

3. Duffin, R. J., E. L. Peterson, and C. M. Zener. Geometric programming - theory and application. New York, New York, John Wiley and Sons, Inc. c1967.

4. Fiacco, A. V. and G. P. McCormick. Nonlinear programming: sequential unconstrained minimization techniques. New York, New York, John Wiley and Sons, Inc. c1968.

5. Hadley, G. Linear programming. Reading, Massachusetts, Addison-Wesley Publishing Co., Inc. 1962.

6. Hadley, G. Nonlinear and dynamic programming. Reading, Massachusetts, Addison-Wesley Publishing Co., Inc. c1964.

7. Heyman, M. and M. Avriel. On a decomposition for a special class of geometric programming problems. Optimization Theory and Applications Journal 3:392-409. 1969.

8. Kuhn, H. W. and A. W. Tucker. Nonlinear programming. Mathematical Statistics and Probability Symposium. 1951.

9. Pascual, Luis D. and Adi Ben-Israel. Constrained maximization of posynomials by geometric programming. Optimization Theory and Applications Journal 5:73-80. 1970.

10. Passy, Ury. Generalization of geometric programming: partial control of linear inventory systems. Unpublished Ph.D. thesis. Stanford, California, Library, Stanford University. 1966.

11. Wilde, D. J. and C. S. Beightler. Foundations of optimizations. Englewood Cliffs, New Jersey, Prentice-Hall, Inc. c1967.

12. Zangwill, W. I. Nonlinear programming, a unified approach. Englewood Cliffs, New Jersey, Prentice-Hall, Inc. c1969.

13. Zoutendijk, G. Methods of feasible directions. Amsterdam, Holland, Elsevier Publishing Co. c1960.

## ACKNOWLEDGMENTS

I would like to acknowledge the considerable assistance from several sources given to me during the period of my graduate work.

A great debt of gratitude is owed to my research advisor, Professor J. K. Sengupta, for his guidance, sound advice, and constant encouragement during the course of my research. I also wish to thank Professor C. G. Maple for his guidance during my graduate studies.

My wife, Jill, deserves special thanks for her patience and biased encouragement which was of great assistance.

# APPENDIX

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION C(30),CB(20)
      DIMENSION A (10,30),IB (20),IBT (30)
      DIMENSION EX (10,30),IIB(20),IIBT (30)
      DIMENSION E (30),DE(30),IBS(30),IT(30)
      DIMENSION ICHECK (30)
      INTEGER OUTPUT
      COMMON /A1/ INPUT,OUTPUT
      COMMON /A2/ A,IBT,IB
      COMMON /A3/ EX,IIBT,IIB
      COMMON /A4/ E,DE,IBS,IT
      COMMON /A5/ C,CB,XXXX,ICOL
      COMMON /A6/ ICHECK
      COMMON /A7/ N,M,M1,MN,KKK
      COMMON /A8/ NT,MT,M1T,MNT
      INPUT = 1
      OUTPUT = 3
C      GEOMETRIC PROGRAM   GARY K.OLESON
C      SOLVED BY USING LINEAR PROGRAMS
C      IMPUT DATA MUST BE A CONSISTENT LINEAR PROGRAM FORMED
C      BY CONVERTING A TIGHT MONOMIAL CONSTRAINED GEOMETRIC
C      PROGRAM.
C      INPUT DATA REQUIRED
C      FIRST CARD   NUMBER OF L.P.CONSTRAINTS
C      SECOND CARD NUMBER OF L.P. COLUMNS
C      AN ARRAY OF SUBSCRIPTS OF EPSILONS ASSOCIATED WITH EACH COLUMN
C      AN ARRAY OF POINTERS TO PAIRED EPSILONS
C      SUBROUTINE INTIAL READS IN THE LINEAR PROGRAM AND SOLVES FOR
C      A BASIC FEASIBLE SOLUTION AND THEN SAVES THIS FEASIBLE
C      SOLUTION FOR USE WITH LINEAR PROGRAM B
      CALL INTIAL
  100 FORMAT (' ',5D19.7)
      WRITE ( OUTPUT ,200) IBS
  200 FORMAT (' ',15I5)
C      COMPUTE SUM OF SQUARES OF ZJ -CJ
      CALL ZJCJ (XX)
    1 YY = XX
```

```
      DO 5 I = 1,NT
    5 WRITE (OUTPUT,101) I,E(I)
  101 FORMAT (' ','E(',I3,')  =',D19.7)
C      FORM LINEAR PROGRAM B USING BASIC FEASIBLE SOLUTION OF A
      CALL FORM
C      COMPUTE ZJ -CJ FOR LINEAR PROGRAM B
      CALL ZJ
C      SOLVE LINEAR PROGRAM B
      CALL LP
C      CHECK IF LINEAR PROGRAM B IS UNBOUNDED,IF SO CALL BOUND
      IF ( ICHECK (1) .NE. 1) GO TO 2
      CALL BOUND
C      COMPUTE FEASIBLE DIRECTION FOR CHANGE OF EPSILONS
    2 CALL DELTA
    6 CONTINUE
C      COMPUTE THE SUM OF SQUARES OF ZJ -CJ
      CALL ZJCJ (XX)
C      IF THE NEW SUM OF SQUARES IS NOT LOWER THAN PREVIOUS
C      DIVIDE THE CHANGE IN EPSILON BY TWO
      IF ( XX.LT. YY) GO TO 4
C  THRU  ...  3 DIVIDES DELTA E BY TWO
      DO 3 I = 1,N
      DE (I) = DE(I) / 2.0D0
    3 E(I) = E(I) - DE(I)
      GO TO 6
C      CHECK FOR TERMINAL CONDITION
    4 IF (XX .GT. 0.1D-15) GO TO 1
 1000 STOP
      END
```

```
      SUBROUTINE INTIAL
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A (10,30),IB (20),IBT (30)
      DIMENSION EX (10,30),IIB(20),IIBT (30)
      DIMENSION E (30),DE(30),IBS(30),IT(30)
      DIMENSION C(30),CB(20)
      INTEGER OUTPUT
      COMMON /A1/ INPUT,OUTPUT
      COMMON /A2/ A,IBT,IB
      COMMON /A3/ EX,IIBT,IIB
      COMMON /A4/ E,DE,IBS,IT
      COMMON /A5/ C,CB,XXXX,ICOL
      COMMON /A7/ N,M,M1,MN,KKK
      COMMON /A8/ NT,MT,M1T,MNT
C  N  EQUALS NUMBER OF MONOMIAL CONSTRAINTS
      READ (INPUT,7) M
C  M  EQUALS NUMBER OF CONVERTED PRIMAL CONSTRAINTS
      READ ( INPUT,7) N
C  IBS (I) CONTAINS SUBSCRIPT OF EPSILON ASSOSIATED WITH MONOMIAL
      READ (INPUT,7) (IBS(I),I = 1,N)
C                    ZERO IF EPSILON (I) IS RESTRICTED TO .GT. 0
C                    -1  IF EPSILON (I) IS RESTRICTED TO .LE. 1
      READ ( INPUT,7) (IT (I),I = 1,N)
C  M + N = SIZE OF LINEAR PROGRAM A WITH ARTIFICAL VARIABLES ADDED
      MN = M + N
      L = N + 1
C  THRU  ...  2 INTIALIZES ARRAY TO ZERO
      DO 1 I = L,MN
      IBS (I) = 0
    1 IT (I) = 0
      M1 = M + 1
      DO 2 I = 1,MN
      IBT (I) = 0
      E ( I) = 0.0D0
      DE (I) = 0.0D0
    2 C (I) = 0.0D0
C  THRU  ...  3 INTIALIZES ARTIFIAL VARIABLES AND THEIR COST FUNCTIONS
```

```
      DO 3 I = 1,M1
      IB (I) = L
      IBT (L) = I
      CB (I) = - 1000.0D0
      C (L) = - 1000.0D0
      DO 4 J = 1,MN
    4 A ( I,J) = 0.0D0
      A (I,L) = 1.0D0
    3 L = L + 1
C   THRU  ...  5 READ IN LINEAR PROGRAM A ARRAY
C                EACH COLUMN CORRESPONS TO A GEOMETRIC PROGRAM
C                MONOMIAL CONSTRAINT
C                EACH ROW CORRESPONDS TO A PRIMAL VARIABLE
      DO 5 I = 1,M
    5 READ ( INPUT,6) ( A(I,J),J = 1,N)
    6 FORMAT (16D5.1)
C   ICOL IS THE COLUMN OF NORMAL C VALUE USED IN NORMALIZATION
C   IVAR IS VARIABLE USED TO NORMALIZE L.P.'S COEFFICIENTS TO ONE
      READ ( INPUT,7) IVAR,ICOL
    7 FORMAT ( 15I5)
C   THRU  ...  8 NORMALIZE CONSTRAINT COSTS TO ONE
      XXXX = DLOG ( A (IVAR,ICOL))
      DO 8 I = 2,N
    8 A ( IVAR,I) = DLOG ( A(IVAR,I))/XXXX
      KKK  = 0
C   THRU  ... 9 INTIALIZES ALL PAIRED EPSILON TO 0.5
C               INTIALIZES ALL OTHERS TO ONE
C               KKK IS TOTAL NUMBER OF EPSILONS USED
      DO 9 I = 1,N
      K = IBS (I)
      IF ( KKK .LT. K) KKK = K
      IF ( K     .EQ. 0) GO TO 9
      E (I) = 0.5D0
      IF ( IT (I) .LE. 0) E (I) = 1.0D0
    9 CONTINUE
C   SUBROUTINE COST COMPUTES COST COEFFICIENTS DEPENDENT ON
      CALL  COST
```

```
C  SUBROUTINE ZJ CALCULATES Z AND Z(J) -C(J) FOR LINEAR PROGRAM A
      CALL ZJ
      WRITE ( OUTPUT,104)
  104 FORMAT (' ',' NORMALIZED INPUT MATRIX')
      DO 14 I = 1,M1
   14 WRITE (OUTPUT, 105) I,(A(I,J),J= 1,MN)
  105 FORMAT (' ','ROW NUMBER OF LINEAR PROGRAM ',I10,20(/5D19.7))
      WRITE ( OUTPUT ,106) (IB(I),I =1,M)
  106 FORMAT(' ','BASIC COLUMNS  ARE',15I5)
      DO 15 I = 1,N
   15 WRITE (OUTPUT,107) I, C(I)
  107 FORMAT (' ','COST COEFFICIENT (',I4,') =',D19.7)
      WRITE ( OUTPUT ,100)
  100 FORMAT (' ','THE SUBSCRIPTS OF EPSILON FOR EACH COLUMN ARE')
      DO 12 I = 1,N
   12 WRITE ( OUTPUT,101) I,IBS(I)
  101 FORMAT (' ','SUBSCRIPT (',I4,') = ',I4)
      WRITE (OUTPUT,102)
  102 FORMAT (' ','POINTERS FOR EPSILON')
      DO 13 I = 1,N
   13 WRITE (OUTPUT,103) I,IT(I)
  103 FORMAT (' ','POINTER (',I4,' ) = ',I4)
C  SUBROUTINE LP SOLVES THE LINEAR PROGRAM
      CALL LP
C  THRU ...  11 SAVES THE BASIC FEASIBLE SOLUTION FO LINEAR PROGRAM A
C       IIBT (J) CONTAINS POINTER TO ROW IF BASIC
C                                  ZERO IF NON BASIC
C      IIB (I) CONTAINS POINTER TO BASIC COLUMN
C  THE REMAINDER SAVES PARAMETERS FOR  LINEAR PROGRAM A
C               INITIALIZES PARAMETER FOR LINEAR PROGRAM B
C          L.P. A       L.P. B
C            N            NT     NUMBER OF COLUMNS
C            M            MT     NUMBER OF VARIABLES
C            M1           M1T    INDEX OF ROW OF Z(J) -C(J)
C            MN           MNT    COLUMN SIZE WITH ARTIFICAL VARIALBLES
      DO 10 J = 1,M
      IIB (J) = IB(J)
```

```
      DO 10 I = 1, MN
   10 EX    (J,I) = A(J,I)
      DO 11 I = 1,MN
   11 IIBT (I) = IBT(I)
      MT = M
      NT = N
      M1T = M1
      MNT = MN
      M = N - 1 - M
      M1 = M + 1
      N = KKK* 2 + 1
      MN = N
      RETURN
      END
```

```
      SUBROUTINE ZJCJ (XX)
C  SUBROUTINE ZJCJ (XX) CALCULATES COST COEFFICIENTS DEPENDENT
C               ON EPSILON
C                 USES THE COST COEFFICIENTS TO COMPUTE Z(J) -C(J)
C                 COMPUTES THE SUM OF SQUARES OF Z(J) -C(J)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION EX (10,30),IIB(20),IIBT (30)
      DIMENSION E (30),DE(30),IBS(30),IT(30)
      DIMENSION C(30),CB(20)
      INTEGER OUTPUT
      COMMON /A1/ INPUT,OUTPUT
      COMMON /A3/ EX,IIBT,IIB
      COMMON /A4/ E,DE,IBS,IT
      COMMON /A5/ C,CB,XXXX,ICOL
      COMMON /A8/ NT,MT,M1T,MNT
      DO 1 I = 1,NT
      C (I) = 0.0D0
      IF ( IBS (I) .EQ. 0) GO TO 1
      C(I) = - DLOG (E(I))
    1 CONTINUE
      C (ICOL) = -XXXX
      DO 2 I = 1,MT
    2 CB (I) = C (IIB(I))
      DO 3 J = 1,NT
      EX (M1T,J) = -C(J)
      DO 3 I = 1,MT
    3 EX (M1T,J) = EX (M1T,J) + CB(I)*EX(I,J)
      XX = 0.0D0
      DO 4 J = 2,NT
    4 XX = XX + EX(M1T,J)*EX(M1T,J)
      WRITE (OUTPUT,5) XX
    5 FORMAT (' ','THE SUM OF SQUARES OF ZJ -CJ  =',D19.9)
      DO 6 I = 1,NT
    6 WRITE (OUTPUT,101)I,I,EX(M1T,I)
  101 FORMAT (' ','Z(',I3,') -C(',I3,')',D19.7)
      RETURN
      END
```

```
      SUBROUTINE COST
C  SUBROUTINE COST USED THE EPSILONS TO CALCULATE COST COEFFICENTS
C                EPSILON AND INTIALIZES THE BASIC COST VECTOR
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A (10,30),IB (20),IBT (30)
      DIMENSION E (30),DE(30),IBS(30),IT(30)
      DIMENSION C(30),CB(20)
      INTEGER OUTPUT
      COMMON /A1/ INPUT,OUTPUT
      COMMON /A2/ A,IBT,IB
      COMMON /A4/ E,DE,IBS,IT
      COMMON /A5/ C,CB,XXXX,ICOL
      COMMON /A7/ N,M,M1,MN,KKK
      COMMON /A8/ NT,MT,M1T,MNT
      DO 1 I = 1,N
      C (I) = 0.0D0
      IF ( IBS (I) .EQ. 0) GO TO 1
      C (I) = -DLOG (E(I))
    1 CONTINUE
      DO 2  I = 1,M
    2 CB (I) = C(  IB (I))
      C (ICOL) = -XXXX
      RETURN
      END
```

```
      SUBROUTINE BOUND
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION C(30),CB(20)
      DIMENSION A (10,30),IB (20),IBT (30)
      DIMENSION ICHECK (30)
      INTEGER OUTPUT
      COMMON /A1/ INPUT,OUTPUT
      COMMON /A2/ A,IBT,IB
      COMMON /A6/ ICHECK
      COMMON /A7/ N,M,M1,MN,KKK
C   SUBROUTINE BOUND USES AN UNBOUNDED SOLUTION OF LINEAR PROBRAM B
C      .I.E. ICHECK (1) = 1
      K = 0
      DO 1 I = 2,N
      IF ( ICHECK (I) .NE. 1) GO TO 1
C   THRU   ...  2  CHECKS FOR ALL DELTA EPSILONS = ZERO
      DO 2 J = 1,M
      IF ( A (J,1) .GT. 0.1D-6) GO TO 3
    2 CONTINUE
      K = K + 1
      IBT (I) = K + M1
      IB ( K + M1) = I
      LL = I - KKK
      IF ( LL.GT. 1) IBT (LL) = 0
      IF ( LL.LT. 0) IBT ( I + KKK ) =0
C   IF ZERO ONE UNIT OF UNBOUNDED SOLUTION IS ADDED
      A ( K + M1,1) = 1.0D0
C   THRU  ...  4 DELTA EPSILONS ARE ADJUSTED BY UNBOUNDED COLUMN
      DO 4 J = 1,M
    4 A (J,1) = A (J,1) - A (J,I)
C   SUBROUTINE ZJ COMPUTES NEW ZJ -CJ
      CALL ZJ
      ICHECK (I) = 0
C   REMOVES MARK OF UNBOUNDED VARIABLE
    1 CONTINUE
    3 RETURN
      END
```

```
      SUBROUTINE FORM
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION C(30),CB(2C)
      DIMENSION A (10,30),IB (20),IBT (30)
      DIMENSION EX (10,30),IIB(20),IIBT (30)
      DIMENSION E (30),DE(30),IBS(30),IT(30)
      INTEGER OUTPUT
      COMMON /A1/ INPUT,OUTPUT
      COMMON /A2/ A,IBT,IB
      COMMON /A3/ EX,IIBT,IIB
      COMMON /A4/ E,DE,IBS,IT
      COMMON /A5/ C,CB,XXXX,ICOL
      COMMON /A7/ N,M,M1,MN,KKK
      COMMON /A8/ NT,MT,M1T,MNT
C  SUBROUTINE FORM SETS UP LINEAR PROGRAM TO COMPUTE DELTA EPSILON
C  THRU   ...  2 INITIALIZES ARRAYS
      DO 1 I = 1,M
      CB (I) = 0.0D0
    1 IB (I) = 0
      DO 2 J = 1,MN
      IBT (J) = 0
      C (J) = 0.0D0
      DO 2 I = 1,M1
    2 A ( I,J) = 0.0D0
C  THRU   ...3 COMPUTE THE PARTIAL DERIVATIVE OF C(J)
      DO 3 I= 2,NT
      KK = IBS (I)
      IF ( KK.EQ. 0 ) GO TO 3
      DE (I) = - 1.0D0/E (I)
      IF ( IT (I) .EQ. 0 ) GO TO 3
      IF ( IT (I) .LT. I) DE (I) = - DE(I)
    3 CONTINUE
      K = 0
C  THRU  ...  4 SETS UP KKK COLUMNS OF L.P.  B ONE FOR EACH EPSILON
      DO 4 J = 2,NT
      IF ( IIBT (J) .NE. 0) GO TO 4
      K = K + 1
```

```
      A (K,1) = -EX (MJT,J)
      KK = IBS (J)
      IF ( KK.NE. 0 ) A (K,KK+ 1 ) = A(K,KK+ 1) - DE(J)
      DO 5 I = 1,MT
      KK = IBS (IIB(I))
      IF ( KK.NE. 0) A(K,KK+1) = A(K,KK+1) + EX(I,J)*DE(IIB(I))
    5 CONTINUE
    4 CONTINUE
C   THRU  ...  6 ADDS COLUMNS TO INSURE POSITIVE DELTA EPSILONS
      DO 6 I = 1,KKK
      KK = I + KKK  + 1
      DO 6 J = 1,M
    6 A (J,KK) = - A(J,I + 1)
C  THRU  ...  7 FORCE INTIAL SOLUTION TO BE BASIC FEASIBLE
      DO 7 I = 1,M
      IF ( A(I,1) .GT. 0.0D0) GO TO 7
      DO 8 J = 1,N
    8 A(I,J) = -A(I,J)
    7 CONTINUE
      J = N
C  THRU  ...  9 ADDS LARGE NEGATIVE COST COEFFICIENTS TO INITIAL SOLUTION
      DO 9 I = 1,M
      J = J + 1
      CB (I ) = - 10000.0D0
      C (J) = -10000.0D0
      IBT (J) = I
    9 IB (I) = J
C  THRU  ... 10 USE L.P. AS OBJECTIVE FUNCTION COST PARTIALS
C               FOR FORMING OBJECTIVE FUNCTION FOR L.P. B
      DO 10 I = 1,MT
      KK = IBS(IIB(I))
      IF (KK.EQ. 0) GO TO 10
      KK = KK + 1
      C (KK) = C(KK) -EX(I,1)*DE(IIB(I))
      C(KK + KKK) = - C(KK)
   10 CONTINUE
      RETURN
```

```
      SUBROUTINE LP
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A (10,30),IB (20),IBT (30)
      DIMENSION C(30),CB(20)
      DIMENSION ICHECK (30)
      INTEGER OUTPUT
      COMMON /A1/ INPUT,OUTPUT
      COMMON /A2/ A,IBT,IB
      COMMON /A5/ C,CB,XXXX,ICOL
      COMMON /A6/ ICHECK
      COMMON /A7/ N,M,M1,MN,KKK
C  SUBROUTINE LP IS A STANDARD SIMPLEX METHOF OF SOLVING L.P.'S
C  THRU   ... 60 INITIALIZES UNBOUNDED CHECK
   65 DO 60 I = 1 ,N
   60 ICHECK (I) = 0
C     SELECTS COLUMN TO ENTER BASIS
   70 IJ= 0
      X =-0.1D-10
C  THRU  ... 51 FINDS MOST NEGATIVE Z(J) - C(J)
      DO 51  I= 2,N
      IF (  A (M1,I) .GE. X) GO TO 51
      IF ( ICHECK (I).EQ.1 ) GO TO 51
      IJ = I
      X  =  A (M1,I)
   51 CONTINUE
      IF (IJ .EQ. 0)  GO TO 100
      JJ = IJ
      K = 0
C  SUBROUTINE SASIS TRIES TO ADD JJ COLUMN TO BASIS
      CALL BASIS ( JJ,K)
      IF (K.NE. 0) GO TO 65
C  YES CONTINUE ON
C  NO MARK AS UNBOUNDED AN TRY OTHERS
      ICHECK (JJ) = 1
      ICHECK (1) = 1
      GO TO 70
  100 CONTINUE
```

```
      DO 14 I = 1,M1
   14 WRITE (OUTPUT, 105) I,(A(I,J),J= 1,MN)
  105 FORMAT (' ','ROW NUMBER OF LINEAR PROGRAM ',I10,20(/5D19.7))
      WRITE ( OUTPUT ,106) (IB(I),I =1,M)
  106 FORMAT(' ','BASIC COLUMNS  ARE',15I5)
      DO 15 I = 1,N
   15 WRITE (OUTPUT,107) I, C(I)
  107 FORMAT (' ','COST COEFFICIENT (',I4,') =',D19.7)
      RETURN
      END
```

```
      SUBROUTINE ZJ
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION C(30),CB(20)
      DIMENSION A (10,30),IB (20),IBT (30)
      INTEGER OUTPUT
      COMMON /A1/ INPUT,OUTPUT
      COMMON /A2/ A,IBT,IB
      COMMON /A5/ C,CB,XXXX,ICOL
      COMMON /A7/ N,M,M1,MN,KKK
C   SUBROUTINE ZJ COMPUTES Z(J) - C(J) GIVEN COST COEFFICIENTS OF LINEAR
C                PROGRAM
      DO 55  J= 1,N
      A(M1,J) = - C(J)
      DO 55  I= 1,M
   55 A(M1,J) = A(M1,J) +  CB(I)* A(I,J)
      RETURN
      END
```

```
      SUBROUTINE BASIS (JJ,II)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION C(30),CB(20)
      DIMENSION A (10,30),IB (20),IBT (30)
      INTEGER OUTPUT
      COMMON /A1/ INPUT,OUTPUT
      COMMON /A2/ A,IBT,IB
      COMMON /A5/ C,CB,XXXX,ICOL
      COMMON /A7/ N,M,M1,MN,KKK
C     SUBROUTINE BASIS IS STANDARD SIMPLEX TABLEAU CHANGE
      X =-1.0D0
C     THRU  ... 52 FINDS PIVOT ROW
      DO 52  I = 1,M
      IF (A (I,JJ) .LE. 0.1D-7)  GO TO 52
      IF (X .EQ.-1.0D0 )  X = A(I,1) / A (I,JJ)
      IF (X .LT. (A(I,1) / A (I,JJ)))  GO TO  52
      X =  A (I,1) / A(I,JJ)
      II = I
   52 CONTINUE
C     RETURN IF COLUMN JJ CANNOT BE ADDED TO BASIS
      IF (II.EQ. 0 ) GO TO 100
C     CHANGE OF BASIS.
C     MARKS CHANGES IN POINTERS
      I = IB (II)
      IBT(I) = 0
      IBT (JJ ) = II
      IB (II) = JJ
      CB (II) =  C(JJ)
C     CALCULATE  NEW TABLEAU
C     THRU  ... 31 COMPUTES NEW ROWS EXCEPT PIVOT ROW
      DO 31  I= 1,M1
      IF (I.EQ. II)  GO TO 31
      X =  A (I,JJ) / A (II,JJ)
      DO  32   J= 1,MN
   32 A(I,J) = A(I,J) - A(II,J) * X
   31 CONTINUE
      X = A(II,JJ)
```

```
C  THRU ... 35 COMPUTES NEW PIVOT ROW
      DO 35  J= 1,MN
   35 A(II,J) = A (II,J) / X
      DO 40 I = 1,M1
   40 A(I,JJ) = 0.0D0
C  THRU  ... 505 CLEAN -UP PROCEDURE
      DO 505 I = 1 ,N
      IF ( DABS (A(M1,I)) .LE. 1.0D-10 ) A(M1,I) = 0.0D0
  505 CONTINUE
      A(II,JJ) = 1.0D0
  100 RETURN
      END
```

```
      SUBROUTINE DELTA
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A (10,30),IB (20),IBT (30)
      DIMENSION E (30),DE(30),IBS(30),IT(30)
      INTEGER OUTPUT
      COMMON /A1/ INPUT,OUTPUT
      COMMON /A2/ A,IBT,IB
      COMMON /A4/ E,DE,IBS,IT
      COMMON /A8/ NT,MT,M1T,MNT
      COMMON /A7/ N,M,M1,MN,KKK
C   SUBROUTINE DELTA EXTRACTS A FEASIBLE DIRECTION FOR CHANGE OF
C                EPSILON
C   THRU  ...  3 ZEROES CHANGE
      DO 1 I = 1,NT
    1 DE (I) = 0.0D0
C   THRU  ...  4 SEARCHES FOR BASIC VARIABLE OF DELTA E
      DO 4 L = 1,KKK
C   THRU  ...  3 FINDS SUBSCRIPT OF DELTA EPSILON
      DO 3 I = 2,NT
      IF ( IBS (I) .NE. L ) GO TO 3
      J= I
      GO TO 2
    3 CONTINUE
    2 JJ = L + 1
C   CHECKS OR POSITIVE CHANGE
      IF ( IBT (JJ)  .NE. 0) DE(J) = A (IBT(JJ),1)
C   CHECKS FOR NEGATIVE
      IF ( IBT (JJ + KKK) .NE. 0) DE (J) = -A(IBT(JJ+KKK),1)
C   CHECKS FOR PAIR CHANGE
      IF ( IT (J) .GT. 0) DE (IT(J)) = - DE(J)
    4 CONTINUE
C   THRU  ...  5 CHECKS FOR VIOLATED CONSTRAINTS ON EPSILON
C              IT (I) POSITIVE IF PAIR MUST EQUAL ONE
C                     ZERO  IF ONLY POSITIVE RESTRICTION
C                     NEGATIVE IF 0.LE. 1
      DO 5 I = 1,NT
      IF ( E(I) .LT. 0.1D-6)  GO TO 5
```

```
      X = E(I) + DE (I)
      IF ( X .GT. 1.0D0) GO TO 6
      IF ( X .GT. 0.0D0) GO TO 5
      X = - E(I) * .95D0/DE(I)
      GO TO 7
    6 IF ( IT(I) .EQ. 0) GO TO 5
      X = (1.0D0 - E(I)) * .95D0/ DE(I)
    7 DO 8 J = 2,N
    8 DE(J) = DE(J) * X
    5 CONTINUE
C   COMPUTE NEW SET OF EPSILONS
      DO 9 I = 1,NT
    9 E(I) = E(I) + DE(I)
      RETURN
      END
```